REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
07-03-2002 * MERGEFORMAT	Conference Proceedings	9 September 2002 - 11 September 2002

4. TITLE AND SUBTITLE

1st International Conference on Artificial Immune Systems ICARIS 2002

5a. CONTRACT NUMBER
F61775-02-WF057

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Conference Committee

5d. PROJECT NUMBER

5d. TASK NUMBER

5e. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Kent at Canterbury
University of Kent at Canterbury
Canterbury, Kent CT2 7NF
UK

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

EOARD
PSC 802 BOX 14
FPO 09499-0014

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
CSP 02-5057

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The Final Proceedings for 1st International Conference on Artificial Immune Systems. ICARIS 2002, 9 September 2002 - 11 September 2002

Artifcial immune systems, artificial intelligence, robot control, intrusion detection, data mining, self-repair, machine learning.

15. SUBJECT TERMS

EOARD, Artificial Intelligence, Intelligent Systems, Information Protection

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18, NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Neal D. Glassman
UNCLAS	UNCLAS	UNCLAS	UL	230	19b. TELEPHONE NUMBER *(Include area code)* +44 (0)20 7514 4437

Proceedings of ICARIS 2002

Editors: J. Timmis and P.J. Bentley

SESSION I: APPLICATIONS OF AIS

SESSION II: MEMORY AND AIS

SESSION III: SELF OR NON-SELF?

SESSION IV: CONCEPTUAL PAPERS

SESSION V: LEARNING STRATEGIES

SESSION VI: HYBRIDS AND AUGMENTATIONS

A MULTILAYERED IMMUNE SYSTEM FOR HARDWARE FAULT TOLERANCE WITHIN AN EMBRYONIC ARRAY.

R.O. Canham **A.M. Tyrrell**

Department of Electronics.
The university of York
Heslington, York. UK
roc100@ohm.york.ac.uk

A MULTILAYERED IMMUNE SYSTEM FOR HARDWARE FAULT TOLERANCE WITHIN AN EMBRYONIC ARRAY.

Abstract

Biology demonstrates high levels of fault tolerance in all instances. This paper documents a demonstration system that takes inspiration from the immune system and embryonic processes to acquire some of these fault tolerant properties in hardware circuits. A multi-layer immune system is used as fault detection; a negative selection algorithm is used at a system level to identify non-self states. These are localised by further tests. Reconfiguration of the embryonic array accommodates the faults located. The detector set for the negative selection algorithm is currently derived by hand, although appropriate learning algorithms are identified and commented upon.

1 INTRODUCTION

As systems become more complex it becomes increasingly difficult to provide comprehensive fault testing to determine the validity of the system. Hence faults can remain in a system which can manifest themselves as errors. Furthermore, faults may be introduced into a hardware system from external sources such as electromagnetic interference. Components within a system can die; no transistor will function forever. These faults can ultimately cause a system to fail. The ability of a system to function in the presence of faults, to become fault tolerant, is a continually increasing area of research.

Through millions of years of refinement biology has produced many living creatures that are remarkably fault tolerant. They can survive injury, damage, wear and tear, and are under continual attack from other living entities in the form of infectious pathogens. This paper details a fault tolerant circuit that takes its inspiration from some of the fault tolerance techniques found in biology.

A hardware multilayered artificial immune system is used as a fault detection system upon an embryonic array (Tempesti 1998, Ortega 2000) which can then accommodate the faults. The embryonic array is a homogeneous array of logic units (called cells) that use their location within the array to extract appropriate configuration data. Each cell contains all the configuration details of all cells and hence can perform any cell's function as required.

This is part of the POEtic project which aims to produce a circuit that combines other forms of bio-inspired techniques (POEtic 2002). These include phylogeny (P), the ability of a population of individuals to change from one generation to the next in an evolutionary manner, ontogeny (O), the developmental processes of an organism's growth. Multi-cellular organisms grow from a single cell that contains all the necessary information, the current implementation of which is the embryonic array. Finally epigenesis (E), the learning process of an individual. This can not only be found in the nervous system but also the some areas of the immune system which learn to recognise pathogens.

Ultimately the PEOtic device will be produced in silicon using ASIC (Application Specific Integrated Circuit) fabrication. However, this paper details the initial stages of just the immune system upon an embryonic array. Although only simulated results are given in this paper the system will shortly be implemented in a commercial Field Programmable Gate Array (FPGA).

2 BACKGROUND INFORMATION

2.1 ARTIFICIAL IMMUNE SYSTEMS

The immune system found in higher organisms is a multilayered, distributed system that is robust and can identify numerous pathogens and other harmful effects. Many of the properties found in such a system would be most advantageous in many computer and other systems. Artificial immune systems do just this. They have been applied to many different application areas, such as: optimisation techniques (Hajela 1999, Endo

1998), novel implementations of neural networks (Hoffmann 1986), anomaly detection (Kim 1999, Forrest 1997), pattern recognition (Hunt 1996, Dasgupta 1999) inductive problem solving (Slavov 1998, Nikolaev 1999) control (Ishiguro 1997), industrial process monitoring (Ishiguru 1994, Ishida 1993), fault tolerant software (Xanthakis 1996) and hardware fault tolerance (immunotronics) (Bradley 2000a, 2000b, 2001, 2002). Further information, surveys and reviews can be found in Dasgupat and Attoh-Okine (Dasgupta 1997), de Castro (Castro 1999) and de Castro and Von Zuben (Castro 2000).

Of the many algorithms and systems researched, many make use of the negative selection algorithm. Developed by Forrest et al. (Forrest 1994), the negative selection algorithm is based upon the detection of non-self from self, as found within the immune system. Various immune cell types (such as lymphocytes) have receptors that allow them to bind to specific sets of proteins. The maturation of each lymphocyte cell involves the presentation of proteins that are naturally present within the body (self). Lymphocytes that bind with them are destroyed. Hence, when released within the body, binding to a protein indicates it is non-self and may be a harmful pathogen. See Janeway (1999), Kimball (2002), de Castro (1999) and Alberts (1994) for more details of immune systems. Forrest uses a string to represent the systems state; partial matching of these strings is used to distinguish between self and non-self. This can be considered similar to the binding of some lymphocytes. The negative selection algorithm can be summarised as follows:

- A set of self strings, *S*, are defined. Each is of a length, *l,* of a finite alphabet. The set of strings can be gathered during operation in an application dependent manner to describe the state of the application.
- A set of detectors, *R*, is generated which fails to partially match any member of the self set, *S*. The partial match used by Forrest is defined as a match of *c* contiguous bits within the string's length.
- The state of the device under test is monitored. Under normal operation this would generate a member of the self set, *S*. However, an abnormal process may generate a non-member of *S* which can be matched by a member of the detector set R.

The detector sets have been generated by a random process, more efficient algorithms (D'haeseleer 1995) and evolutionary processes (Kim 1999), including library selection (Hunt 1996). Detectors that match self are destroyed in a similar manner to the biological immune system.

2.2 IMMUNOTRONICS

All the applications listed are software implementations of an artificial immune system. The only hardware implementation found to date is Immunotronics (immune + electronics) by Bradley and Tyrrell (Bradley 2000a, 2000b, 2001, 2002). This makes use of a negative selection algorithm to identify faults within a hardware circuit, specifically a finite state machine. The current state, next state and the current inputs are used to define the current transition of the machine. Some of the transitions are not present in normal, error free operation and so are considered as non-self. Identification of such non-self indicates the presents of an error.

A 4 bit BCD counter implemented upon a Xilinx Virtex FPGA was immunised. The detector set was generated offline using both random techniques and the greedy algorithm (D'haeseleer 1995). The detectors themselves were implemented using a content addressable memory (CAM) (Xilinx 1999). This allows all the elements within the memory to be compared with the current state very quickly. Partial matching based upon the number of contiguous bits was employed. Experimental results showed that a fault coverage of 93% could be achieved with 103 detectors, which constitutes 10% of all possible error, and therefore detector, states.

This is very impressive but requires a CAM of 103 words, each 10 bits in width. This is vast in size compared to the counter that was being immunised. However, the size of the detectors is not dependant on the complexity of the circuit being immunised. It might be considered that the counter is the wrong granularity. Also a biological immune system never tries to provide a fault free functionality. The underlying system requires some inherent fault tolerance; indeed the immune system will typically kill off infected cells. The biological entity can easily accommodate this due to the vast quantity of redundancy.

It is therefore more appropriate to use an immune system at a system, or sub-system level. It would not now be used to identify every error, but unusual situations that indicate something erroneous has occurred. If a robot controller is considered, the immune system would provide monitoring for situations that should not occur. This may include situations such as a robot that is heading for an object.

2.3 EMBRYONICS

All multi-cellular organisms start life as a single cell. This cell divides repeatedly to generate numerous identical copies of itself. Each cell contains all the information necessary to create the entire entity – the genotype. As the number of cells increases differentiation takes place; different cells start to change to provide different, specialised functionality. The appropriate section of the genotype (the appropriate gene or genes) is selected based upon the cell's position as well as other factors.

Embryonics (embryo + electronics) is inspired by the cloning and differentiation of cells within multi-cellular organisms to generate electronic circuits with some of the properties of such organisms (Tempesti 1998, Ortega 2000, Prodan 2001, Stauffer 2001). An

embryonic array consists of a homogeneous array of *cells,* each containing the full specification of the device, together with a processing element and control. The coordinate position of each cell is calculated dependent upon its neighbours; this is used to perform the appropriate section of the genotype. Figure 1 gives an example of some of the major elements of a generic array. Implementations to date typically use a very simple functional unit (a two input multiplexer), although a number of these units have been included in a cell. These sub-sections of the cell have been termed molecules.

Errors within the array are accommodated by killing the particular cell. The routing becomes transparent and the coordinate system no longer increments, thus the next cell takes over the functionality of the faulty cell. The array contains spare cells that are not utilised until a fault occurs.

To maintain cellular alignment removing the row or column that contains the error is typical (see Figure 2). It should be reiterated that no configuration data has to be recalculated or moved; just the change in coordinate is all that is required for the cells to reconfigure themselves. Fault avoidance has also been performed at the molecular level. As many faults as there are spare cells can be handled.

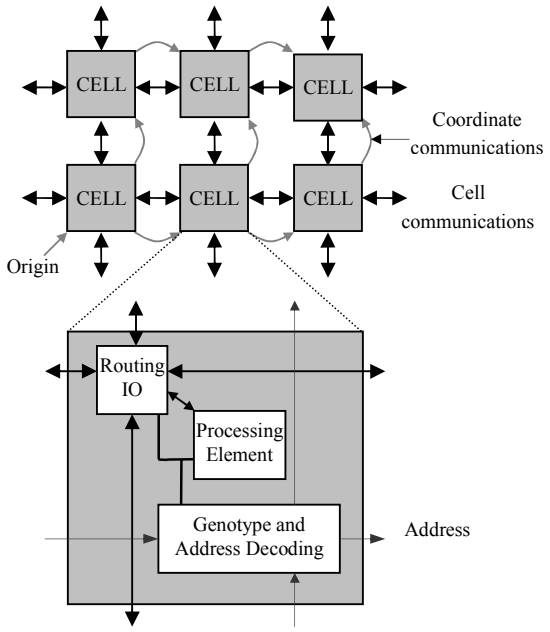


Figure 1: Embryonic Array

Implementations to date use replication of the functional units and a comparator to provide built in self tests (BISTS). At present it is assumed that the routing and control are fault free.

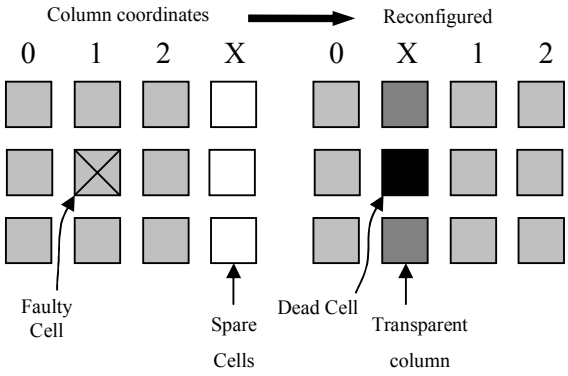


Figure 2: Accommodation of Fault

3 THE SYSTEM

3.1 OVERVIEW

The new system proposed uses a number of *layers* to provide fault detection using the immune system for inspiration. The top level is a negative selection algorithm that learns and is analogous to the acquired immune system. This can work at either a system or subsystem level and monitors the state of the system for non-self. When non-self is identified the whole system could be relocated by killing the cells within the embryonic array. However, such resources are rarely available and so a second layer is used to localise the fault. This could take many forms; here a number of configurations are loaded which perform a number of tests. The device could be tested in sections with a reduced functionality, or in a roving self-test area (STAR) (Emmert 2000, Abramovic 2001). In this example the device is briefly placed in a safe state and taken offline while the testing takes place, in a similar manner to Sundararajan (2001). Cells that have errors are identified and killed. These tests do not learn and hence could be considered to be mapped to the biological innate immune system.

A simple embryonic array was generated to provide the ability to reconfigure around faults that were found. The criteria of the array was to provide suitable functionality to implement the immune system and to enable it to be implemented on a commercial FPGA. Therefore it may bear little resemblance to the embryonic array that will be produced in the final POEtic tissue. The immune system will provide testing for many of the areas that current implementations of embryonics assume to be fault free.

A simple example application is used to demonstrate the system. This takes the form of the simplest of robot controllers which has three single bit inputs and two single bit outputs; the inputs represent the detection of an object to the robot's left, front and right. The outputs represent signals to the robot's left and right motors. Hence a signal on the left motor will turn the device to the right, both motors will cause it to travel forward and

a signal on the right motor will turn the robot to the left. This is shown in Figure 3.

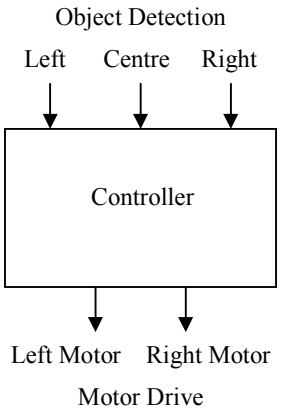


Figure 3: Test System

The nature of the controller is not important; in this demonstration it took the form of a simple lookup table which was driven by a linear feedback shift register to generate a random, complete set of inputs.

When the outputs are considered with the inputs there are a number of states that would be considered as normal (self-states) and those that would be considered as abnormal (non-self). An example of this would be the detection of a object ahead with both motors driving the robot into it. A complete set of these non-self states is given in Table 1. In this simple example the states were determined by hand, although an automated learning capability is ideal. Details of learning processes are given in section 5.1.

Input states			Outputs	
Left	Ahead	Right	Left	Right
0	0	1	1	0
0	1	0	1	1
0	1	1	1	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1

Table 1: Non-self Sates

3.2 THE EMBRYONIC ARRAY

The embryonic array is based upon the hardware-orientated arrays produced by Ortega and Tyrrell (Ortega 2000). However, many changes were required. Each cell of the array is constructed from a preset number of molecules. Dedicated hardware is used to generate other cell functionality, such as address calculation, gene selection and maintaining the cell's state. Each molecule contains a four input lookup table (LUT) and a D-type flip-flop. A three bit bus connects each molecule to its four nearest neighbours via a switch block that provides full connectivity. No direct connections (that by-passed the switch block) are present between molecules. The output of the molecule's LUT or flip-flop can also be connected to any of the switch block's outputs. Each input of the LUT can be connected to any of the switch block's inputs. See Figure 4. There is no distinction between neighbouring molecules within the same cell and the neighbouring cells.

The configuration of each molecule is controlled by a configuration register; each bit of this register is implemented using a 16 bit shift register / 4 bit LUT (i.e. the basic LUT of many FPGAs). Hence, each bit has up to 16 possible options to be selected from. All configuration registers of a cell are linked into a continuous shift register which will contain the configuration of all the cells, i.e. the genotype of the array. Therefore, up to 16 different configurations, or genes, can be stored and the appropriate gene selected dependent upon the cell's location and state. The genotype is distributed, in parallel to each cell, to configure the whole array.

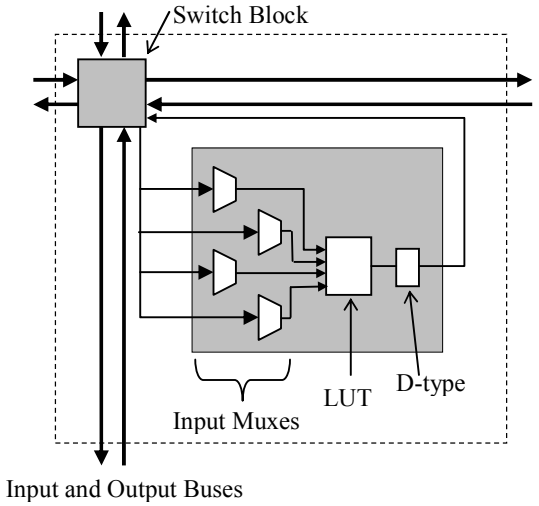


Figure 4: Molecule Block Diagram. Note the output can bypass the D-type if required.

The coordinates for each cell are calculated based upon its neighbours below and to its left; each cell increments the address in the x and y axis which is then propagated on. A mapping, from the coordinates to the selection of the actual gene used, is provided via a lookup table. The mapping data is present at the head of the genotype and allows the same gene to be present in more than one, or no cells, as required.

Faults are accommodated by typically killing the entire column in which the faulty cell is present. A gene is

selected that sets all the switch blocks to be transparent and the cell coordinates are no longer incremented. However, it is also possible to kill a row. This is necessary if the cell will not die appropriately or there is a fault present in the transparency of the cell. This allows a cell to be completely removed, no matter the degree of damage (see Figure 5). Each molecule has the ability to use its functional output to indicate that the cell should die and the row, or column should go transparent. This is controlled via another configuration bit.

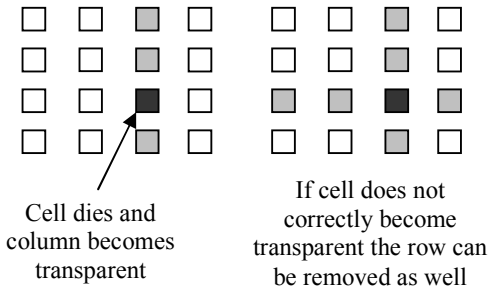


Figure 5: Possible reconfigurations to avoid faults

Control of the cell's state is achieved by a small state machine.

3.3 THE ARTIFICIAL IMMUNE SYSTEM

As stated, the fault detection takes the form of a number of layers. The top-most layer is the implementation of a negative selection algorithm.

3.3.1 The Negative Selection Algorithm and its Implementation

Using the negative selection algorithm to identify unusual situations at system level reduces the number of error states dramatically. In this example the number is reduced to such an extent that there are more self states than non-self states. This makes partial matching between detectors and the current state no longer possible, since the partial match is more likely to match with another self state than non-self state. The number of holes becomes too large to be practical, even with appropriate changes in the state representations (Hofmeyr 1999). The nature of the partial match within the negative selection algorithm (and in biological systems) is very powerful and, some may argue, a key component of the system. It allows a greatly reduced set of detectors to identify a wider range of pathogens; the clonal selection process (the process by which the lymphocytes are chosen for replication) also requires a degree of match to function correctly. However, when implemented in hardware there are a number of constraints and restrictions that prevent full advantage of the partial matching. The luxury of complex software algorithms is not available. Hardware systems tend to maximise the hardware present and so non-used states or conditions are minimised. Hence, most situations would not produce the very high ratios of self to non-self that occur in biology and provide the great power of partial matching.

The reduced number of non-self states, and the small number of bits in this example allows for complete detector sets which are still very small. With a detector of only five bits a complete comparison can be implemented in a simple five bit lookup table (implemented in three 4 bit LUTs). This compares favourably with the typical CAM implementation used in Bradley (2001) which required a LUT for every two bits of a detector and used over 100 detectors.

The system still identifies non-self states which is the fundamental aspect of the algorithm.

The implementation of the detectors is very simple as can be seen in Figure 6. The three LUTs provide the logic necessary to produce an error output for the non-self states shown in Table 1.

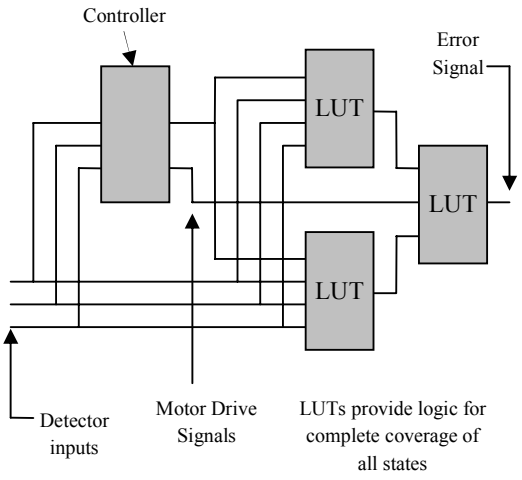


Figure 6: System block diagram

The array simulated used 10 molecules in a cell (two columns of five) which allowed the linear feedback shift register and LUTs for the controller to be implemented in one cell and the detectors in a second cell.

3.3.2 Innate Layer

When a non-self state is identified the error must be located and accommodated. This is achieved by reconfiguring the device and performing test patterns at a low level. The number and type of tests performed can generate an exhaustive test sequence.

Replication of the functional units within the embryonic array has been employed in previous implementations. This provides fault tolerance of the functional units with acceptable overheads and could be included as an additional layer (in biology, cells continually check their functionality and will die by apoptosis if an error is located, Kimball 2002). The tests would therefore concentrate on the routine and switching of the array, which is much harder to produce by replication. These would be similar in nature to the offline test detailed in

Sundararajan (2001) or those performed by the roving star (Emmert 2000, Abramovic 2001). In essence each configuration of a switch block is induced using a suitable test pattern generator and tester, as shown in Figure 7. A large number of tests would be performed in parallel, with the repetition of the same gene in a number of the cells.

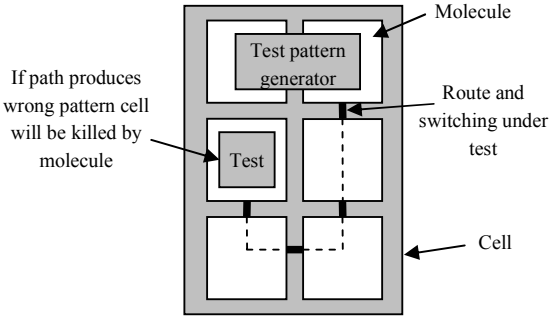


Figure 7: Example test configuration

The tests are typically kept within the cell which is killed by the identification of an error within it. This greatly simplifies any control or localisation needed to reconfigure the device and avoid the fault. Some tests require the data paths between cells to be included; this too causes no complications since the whole column or row is typically avoided.

With an appropriate architecture it would also be possible to test the correct functionality of the cell's control and status and its ability to die. Hence all aspects of the device could be tested.

A control process is necessary to carry out this reconfiguration and error checking. On the final POEtic device a microcontroller will be included within the device which will be utilised. However, this generates a large single point of failure. In an ultimate solution the controller would be implemented within the array itself which then offers all the fault tolerant protection provided. It should be noted that unlike many other FPGA fault tolerant techniques that use reconfiguration to locate and/or avoid faults (such as Blanton 1998) there is no complex reconfiguration calculation required; this is all achieved by the embryonic array. Hence, an integrated controller would be practical. With a reasonable number of test configurations the memory required to store them may become inappropriate. However, the nature of the test genes is such that they are very repetitious and so they could be simply constructed from a number of small subsets of the gene.

Within the simulation of the test application the detector set was replicated within its cell to prevent false negatives. An inconsistent result from the two detector sets causes the immune cell to be killed. Since this example application implements the controller in a single cell, any non-self detected causes that cell to be killed and its column made transparent. However, innate tests are performed to check the correct transparency of the device. Failure of this test results in the appropriate row being made transparent.

This transparency test requires two configurations. These are shown in Figure 8 and Figure 9. Each row (of three cells) is the same. In the first test, coordinate (0,0) contains a gene A, while all the other cells use the gene to make them transparent (as shown in Figure 8). This generates a test pattern which passes the length of the row and is mapped back again. The switching within the test cell is such that the data passes through all horizontal paths and switches to a test molecule at the bottom of the cell. Notice that the test is repeated to prevent an error in the test generating a false negative. A second configuration is required to test the transparency of the end cell, as shown in Figure 9. Gene B is the same as A but it is mirrored in the x axis.

It is necessary for the routing to map the test signals back; this can be achieved in the end cell or externally, as shown.

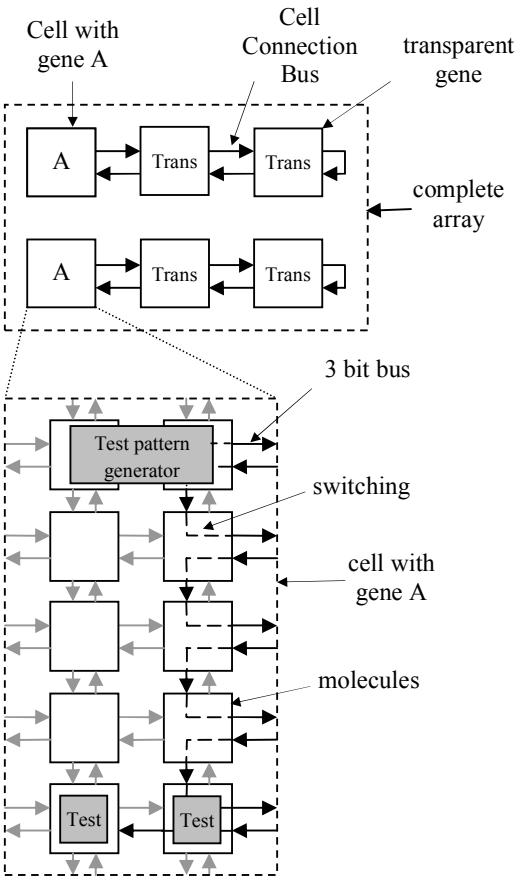


Figure 8: Transparency test

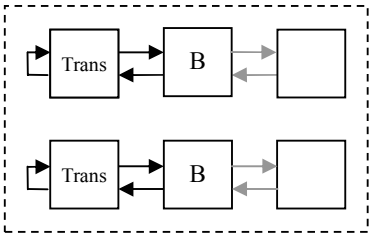


Figure 9: Second transparency test

4 RESULTS AND DISCUSIONS

Figure 10 shows a section of simulation of the circuit. The circuit was written in synthesisable VHDL, optimised for a Xilinx Virtex. Hence the circuit will easily be implemented in hardware. The lower set of traces is an expansion of a section of the upper. Here the inputs and the outputs to the controller can be seen (labelled "controller"). The lower three traces are the detector signals and the upper two traces are the appropriate response for the motor outputs. At time 5500ns a stuck at 1 fault is injected on the output of one of the controller outputs (the upper on the trace). This places the device in an error state and an error signal is generated. This causes cell 0,0 to pass from an OK state, through a brief reconfiguration state, to a dead state. The selected gene goes from 0001 to 0000, the transparent gene. The state of the device goes from normal to test1. This can be seen more clearly in the upper set of traces, showing a larger time scale. Both test conditions are performed before returning to a normal state. The test passes correctly and no further action is performed. Notice that the controller output was on cell 0,0 which, after the testing and reconfiguration, remains at a logic level 0. Cell 1,0 has now taken over the functionality of the controller. The actual output of the device would remain the same.

A similar stuck at fault is injected in the simulation shown in Figure 11. However, the fault is inserted on the connection bus between molecules. This error can not be avoided by a horizontal relocation since the stuck at fault prevents the cell from becoming transparent.

It can be seen that at time 5500ns the fault is injected and as before the cell is killed. The two test configurations are loaded and the tests are performed. This time the second test fails and the Y error line is activated which induces the cell to die and for the row to become transparent. The combined outputs of a number of cells are shown in the lower three traces. Before the error is introduced (and after the configuration) the functionality of the controller is performed by cell 0,0. This is labelled *Output Active* on the figure. After the reconfiguration, cell 1,1 now provides the correct output. Notice that cell 1,0, the cell that was used in the previous example, is no longer used since the y=0 row can not be guaranteed to function correctly.

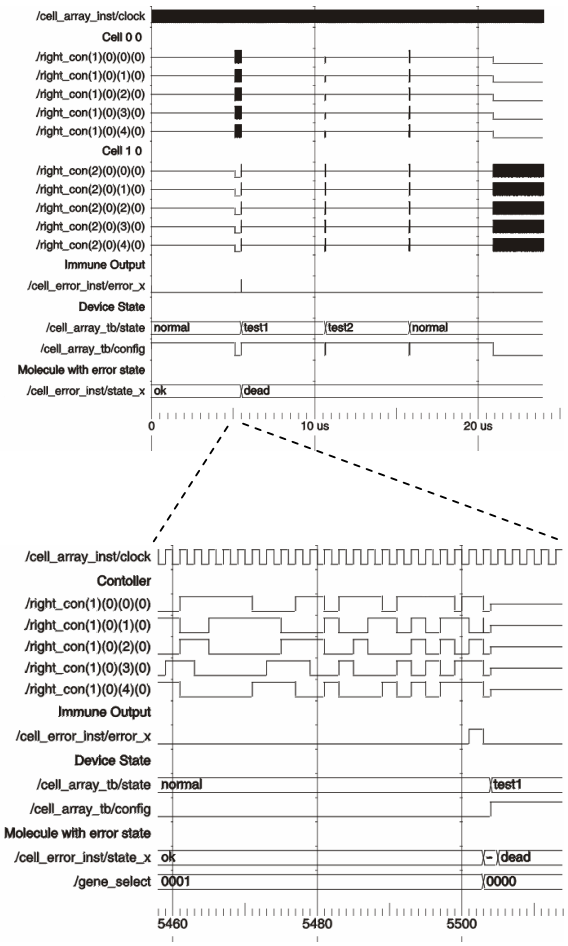


Figure 10: Error injection in molecule function output

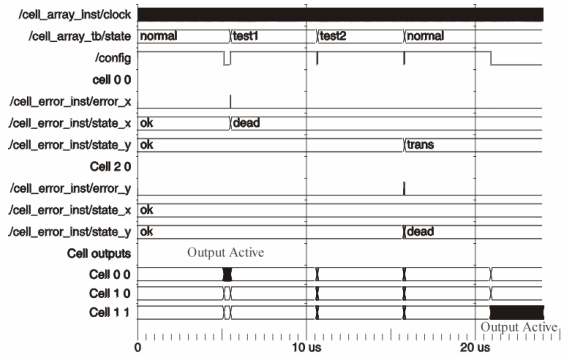


Figure 11: Error injected into connecting bus

5 OTHER CONSIDERATIONS

5.1 LEARNING

In the example simulated the detector set was generated by hand. However, the ability to learn is paramount to the adaptive immune system. During the maturation of B and T cells in a biological immune system, they are exposed to self proteins; those that bind to these are

destroyed. This can be achieved in this application by first setting all the states as error conditions (i.e. filling the LUTs with 1s). A learning period, with fault free operation, is used to present self states to the immune system; each state that occurs is self and this state is removed from the LUT. This type of process is common in many artificial immune systems to generate a detector set. Within a hardware situation there are some difficulties. As stated, the process has to be error free (which is sometimes difficult to guarantee) and all self states have to be demonstrated. This becomes a non-trivial process in a complex system. Adaptive systems (such as those that will populate the final POEtic device) also pose problems.

In biological systems it is not guaranteed that all cells are exposed to all self proteins. Processes exist that allow self binding cells to be accommodated. B cells require co-stimulation by helper T cells (too complex for a hardware system). The state of the immune system also changes the response of a binding cell. If a killer T cell binds to a protein without any other stimulation, it is quite possible that the cell is binding with self and no action is taken. However, if there are other indicators that pathogens are present then the binding cell may well replicate and kill the infected cells. Cell damage (presented to the immune system by antigen presenting cells) is one such indicator of the presence of pathogens. In essence, more that one trigger is required before potentially damaging action is instigated.

This could be emulated by using the knowledge gained from the comprehensive testing process that follows the artificial immune system's detection of non-self. If no errors are found then that detector is probably identifying self and should be destroyed. However, care should be taken to consider a temporary fault. Again taking inspiration from biology, detectors should be periodically created, or more than a single instance of detection is required before action is taken.

Much further work is required to investigate the full potential of this process.

5.2 SCALABILITY

The innate section of the system is not limited by scale. The same tests are typically performed whatever the size of the device; simply more tests are performed in parallel.

The limit of the scalability of the implementation of the negative selection algorithm is the size of the detector set required. This is dependent upon the number of bits that describe each state and the number of error states present. The actual system size is not relevant. If a robot controller is once more considered as an example, then it would be probable that the outputs of any detectors and values of motor controls would contain more bits. However, it would be quite possible to reduce these, with appropriate logic, to produce simple signals that determin features such as if an object was near or that a motor was being driven forward. Much of the extra data are not necessary for the immune system and can be compared to the feature extraction process performed by the major histocompatibility complex in biological immune systems. However, more complex systems with much larger numbers of bits per state can still be easily accommodated. It would no longer be appropriate to have a complete coverage of all states which would complicate the learning algorithm since some decision upon which detectors to include would be required. This is harder to implement in hardware; however, it would not be uncommon for this to be performed in software, offline.

6 CONCLUSIONS

A demonstration of a multilayer artificial immune system implemented within an embryonic array for hardware fault tolerance has been simulated. Using a negative selection algorithm to monitor the system's state for situations that should not occur reduces the number and size of the detector set hence reduces the size of the immune system. A non-learning layer could then be used to localise the fault.

The use of an embryonic array provides an ideal process to avoid the faults located. The immune system adds considerably to the fault tolerance of current implementations of embryonic arrays.

The process currently uses a detector set that is selected by hand; however, learning algorithms have been identified that could be applied. The system also shows promise for scaling to larger systems than that demonstrated.

Acknowledgements

The authors would like to thank other members of the POEtic team for their help and ideas. This project is funded by the Future and Emerging Technologies programme (IST-FET) for the European Community, under grant IST-2000-28027 (POETIC). The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

References

M. Abramovic, J. Emmert and C. Stroud. Roving STARS: An Integrated Approach to On-Line Testing Diagnosis and Fault Tolerance for FPGAs in Adaptive Compuitng Systems. The 3rd NASA/DoD workshop on Evolvable Hardware. Pages 73-92. 2001.

R.D. Blanton, S.C. Goldstein and H. Schmit. Tunable Fault Tolerance via Test and Reconfiguration. *International Fault-Tolerant Computing Symposium.* 1998.

B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts and J. Watson. *Molecular Biology of the Cell.* 3rd Edition. Garland Publishing, New York. 1994.

D.W. Bradley and A.M. Tyrrell. Immunotronics: Hardware Fault Tolerance Inspired by the Immune

System. *Proceedings of the 3rd International Conference on Evolvable Systems. Lecture Notes in Computer Science*, Springer-Verlag. **1801**:11-20. 2000a.

D.W. Bradley, A.M. Tyrrell. Hardware Fault Tolerance: An Immunological Solution. *Proceedings of IEEE Conference on Systems, Man and Cybernetics.* **1**: 107-112. 2000b.

D.W. Bradley, A.M. Tyrrell. Multi-layered Defence Mechanisms: Architecture, Implementation and Demonstration of a Hardware Immune System. *4th International Conference on Evolvable System. Lecture Notes in Computer Science.* **2210**:140–150. 2001.

D.W. Bradley, A.M. Tyrrell. A Hardware Immune System for Benchmark State Machine Error Detection. *Congress on Evolutionary Computation,* 2002.

L.N. de Castro. *Artificial Immune Systems: Part1 – Basic Theory and Applications*. Technical Report TR-DCA 01/99. State University of Campinas. 1999.

L.N. de Castro and F. J. von Zuben. Artificial Immune Systems: Part2 – A Survey of Applications. Technical Report DCA-RT 02/00. State University of Campinas. 2000.

D. Dasgupta and N. Attoh-Okine. Immunity-Based Systems: A Survey. *Proceeding IEEE International Conference on Systems, Man and Cybernetics*. **1**:369-74. 1997.

D. Dasgupta, Y. Cao and C. Yang. An Immunogenetic Approach to Spectra Recognition. Proceedings of GECCO'99. Pages 149-155. 1999.

P. D'haeseleer. *Further Efficient Algorithms for Generating Antibody Strings*. Technical Report of the University of New Mexico. No. CS95-3,11/1/95. 1995.

J. Emmert, C. Stroud, J.Cheatham, A.M. Taylor, P. Kataria and M. Abramovici. Performance Penalty for Fault Tolerance in Roving STARs. *Field-Programming Logic and Applications*. Pages 545-554. 2000.

S. Endo, N. Toma and K. Yamada. *Immune Algorithm for n-TSP*. Proceedings of the IEEE Systems Man and Cybernetics'98. Pages 3844-3849. 1998.

S. Forrest, A.S. Perelson, L. Allen and R. Cherukuri, *Self-nonself discrimination in a computer*. In Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, Los Alamos, CA: IEEE Computer Society Press. Pages 202-12.1994.

S. Forrest, S.A. Hofmeyr and A. Somayaji. Computer Immunology. *Communications of the ACM*. **40**(10); 88-96. 1997.

C.A. Janeway, P. Travers and M. WalPort. *Immuno Biology. The Immune System in Health and Disease.* 4th Ed. Current Biology Publications. London, New York. 1999.

G. W. Hoffmann. A Neural Network Model Based on the Analogy with the Immune System. *Journal of Theoretical Biology*. 122:33-67. 1986.

P. Hajela, and J. S. Yoo . Immune Network Modelling in Design Optimization. In *New Ideas in Optimization*, Editors, D. Corne, M. Dorigo and F. Glover. McGraw Hill, London. Pages 203-215. 1999.

S.A. Hofmeyr and S. Forrest, Architecture for an Artificial Immune System. *Evolutionary Computation*. **7**(1):45-68. 2000.

J. E. Hunt, and D. E. Cooke. Learning Using an Artificial Immune System. *Journal of Network and Computer Applications*. **19**:189-212. 1996.

A. Ishiguro and Y. Watanabe and T. Kondo. A Robot with a Decentralized Consensus-Making Mechanism Based on the Immune System. *Proceedings of Third International Symposium on Autonomous Decentralized Systems.* Pages 231-237. 1997.

Y. Ishida. An Immune Network Model and its Applications to Process Diagnosis. *System and Computer in Japan.* **24**(6); 38-45. 1993.

A. Ishiguru, Y. Wananabe and Y. Uchikawa. Fault Diagnosis of Plant Systems Using Immune Networks. *Proceedings of the 1994 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MIT' 94)*. Pages 34 – 42. 1994.

J. Kim and P. Bentley. Negative selection and niching by an artificial immune system for network intrusion detection. *Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference*. Pages 149-158. 1999.

J.W. Kimball. *Biology*. Web URL: http://www.ultranet.com/~jkimball/BiologyPages/W/Welcome.html. 2002.

N.I. Nikolaev, H. Iba and V. Slavov. Inductive Genetic Programming with Immune Network Dynamics. *Advances in Genetic Programming 3*, MIT Press. Pages 355-376. 1999.

C.Ortega and A.M.Tyrrell A Hardware Implementation of an Embryonic Architecture using Virtex FPGAs. I*n proceedings of the 3rd International Conference on Evolvable Systems. Lecture Notes in Computer Science.* **1801**:155-164. 2000.

L. Prodan, G. Tempesti, D. Mange and A. Stauffer. Embryonics: Artificial Cells Driven by Artificial DNA. *4th International Conference ICES, Lecture Notes in Computer Science.* **2210**:100-110. 2001.

POEtic Project Web Site: http://POEticTissue.org. 2002.

G. Tempesti. *A Self-Repairing Multiplexer-Based FPGA Inspired by Biological Processes.* PhD Thesis. École Polytechnique Fédérale de Lausanne. 1998.

V. Slavov and N.I. Nikolaev. Immune Network Dynamics for Inductive Problem Solving. *Proc of the 5th Conference on Parallel Problem solving from Nature.* Pages 712-721. 1998.

A . Stauffer, D. Mange, G. Tempesti, and C. Teuscher. BioWatch: A Giant Electronic Bio-Inspired Watch. *The*

3rd NASA/DoD Workshop on Evolvable Hardware. Pages 185-192. 2001.

P. Sundararajan and S. McMillan and S. Guccione. Testing FPGA Devices Using JBits. *2001 MAPLD*. 2001.

S. Xanthakis, S. Kararpoulios, R. Pajot and A. Rozz. Immune System and Fault Tolerant Computing. *Artificial Evolution - Lecture Notes in Computer Science.* **1829**:181-197. 1996.

Xilinx. *An Overview of Multiple CAM Designs in Virtex Family Devices.* Application Note 201. URL: http://www.xilinx.com/xapp/xapp201.pdf. 1999.

Extending the Computer Defense Immune System: Network Intrusion Detection with a Multiobjective Evolutionary Programming Approach

Kevin P. Anchor[*], **Jesse B. Zydallis, Gregg H. Gunsch, Gary B. Lamont**
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
2950 P Street, Bldg 640, Dayton, OH 45433-7765
{first.last}@afit.edu

Abstract

Attacks against computer networks are becoming more sophisticated, with adversaries using new attacks or modifying existing attacks. The research uses two types of multiobjective approaches, lexicographic and Pareto-based, in an evolutionary programming algorithm to develop a new method for detecting such attacks. This development extends the Computer Defense Immune System, an artificial immune system for virus and computer intrusion detection. The approach "vaccinates" the system by evolving antibodies as finite state transducers to detect attacks; this technique may allow the system to detect attacks with features similar to known attacks. Initial testing indicates that the algorithm performs satisfactorily in generating finite state transducers capable of detecting attacks.

1 Introduction

Attacks, or intrusions, against computer systems and networks have become commonplace events. Many intrusion detection systems and other tools are available to help counter the threat of these attacks; however, none of these tools is perfect, and attackers are continually trying to evade detection. This paper presents research into detecting new attacks using a new type of antibody for the Computer Defense Immune System (CDIS). These antibodies, which are implemented as finite state transducers, are created using multiple objectives in an evolutionary programming algorithm.

[*]The views expressed in this article are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

The paper is organized as follows. Section 2 briefly discusses intrusion detection systems, evolutionary programming (EP), and multiobjective evolutionary algorithms (MOEA). Section 3 discusses other CIS-based work in the intrusion detection area. The problem addressed in this paper is defined and discussed in Section 4, and Section 5 discusses the algorithms chosen to solve the problem. The associated tests, results, and analysis are described in Section 6, followed by the conclusions.

2 Background

2.1 Intrusion Detection Systems

An intrusion detection system (IDS) helps detect and identify attacks on a computer system or network. The IDS detects attacks by collecting and analyzing information; this information may consist of network traffic, data from a particular host, or both. This research focuses only on the network traffic, so it is a network-based IDS.

An IDS can also be categorized based on its approach for detecting an attack. The main categories are signature-based, anomaly-based, and compound or hybrid [1]. A signature-based system uses knowledge about an attack, such as the pattern or signature of the attack, to determine whether an attack is occurring. If the system does not recognize an attack pattern, then it assumes the data is acceptable [1]. A main disadvantage to this type of system is that an attack that is not in the knowledge base is not detected. The second technique is known as anomaly-based intrusion detection. This technique uses a model of known good behavior and then detects deviations from this model. Any behavior that does not match the model is assumed to be an intrusion [1]. Thus, the system can detect new attacks because it does not rely on a knowledge base of known attack patterns; instead, it relies on a "knowledge base" of known good behaviors. For this type of system, the model of known good behavior must be accurate or the system generates many false detection warnings. The research sys-

tem discussed in this paper is a compound or hybrid of the two forms, as it uses knowledge about an attack and information based on a partial model of known good network traffic, or "self," to evolve finite state transducers (FSTs) as antibodies that can detect the attack and other similar or related attacks.

Several immune system-based IDSs are discussed in Section 3.

2.2 Evolutionary Programming

At a high-level, the standard EAs are all very similar in that they use a model of the biological process of evolution as a framework for the algorithm. However, each class of algorithm has its own representation, reproductive operators, and selection procedure. These differences explain why some EAs are better suited to certain problems; the differences also show that the evolutionary process can be modeled at many levels and in many ways.

One type of EA is Evolutionary Programming (EP), which is similar in concept to other EAs but differs in the generation of offspring from the parent population members. Typical EP algorithms utilize a mutation operator and generate one offspring for each parent population member without the use of recombination. A standard EP algorithm begins by initializing a population of individuals randomly. This process is meant to generate a wide spread of solutions within the search space. Once the starting population is generated, all of the members are evaluated based on the defined fitness function. The fitness value assigned to each of the population members is necessary for the selection operators that are utilized in a later step. A mutation operator is applied to each of the population members to "move" them throughout the search space. This is the "searching" process that the EP conducts to find the "best" solution. This offspring population of solutions are evaluated and a selection operator is applied over the combined population to determine which members are fit to become the parent population for the next generation. The algorithm terminates after some specified stopping criteria [2].

The research algorithm discussed in Section 5 uses evolutionary programming to generate a new type of antibody for the CDIS.

2.3 Multiobjective Evolutionary Algorithms

A relatively new and increased focus of much research is in Multiobjective Evolutionary Algorithms (MOEA) [3, 4]. This area of the EA field is currently of interest to many researchers due to its applicability to real-world problems. In order to understand the concepts applied in the multiobjective version of our algorithm, some terminology must be defined. The process of finding the global maximum or minimum of a set of functions is referred to as Global Optimization. In general, this formulation must reflect the nature of multiobjective problems (MOP) where there may not be one unique solution but a set of solutions found through the analysis of associated Pareto Optimality Theory. MOPs typically consist of competing objective functions, which may be independent or dependent on each other. Many times MOPs force the decision maker to make a choice, which is essentially a tradeoff, of one solution over another in objective space. MOPs are those problems where the goal is to optimize n objective functions simultaneously. This may involve the maximization of all n functions, the minimization of all n functions or a combination of maximization and minimization of these n functions. The formal definition of an MOP is found in [5].

The solution to an MOP is the set of solutions on the Pareto Front, which represent optimal solutions in the sense that improving the value in one dimension of the objective function vector leads to a degradation in at least one other dimension of the objective function vector. This forces the decision maker to make a tradeoff decision when presented with a number of optimal solutions for the MOP at hand, i.e. the Pareto Front. The decision maker typically chooses only one of the associated Pareto Optimal solutions, $\vec{u} \in \mathcal{PF}^*$, as being the acceptable compromise solution, even though all of the Pareto Optimal solutions are optimal [5].

MOPs typically consist of competing objective functions, which may be independent or dependent on each other. An example of this is a company's quest to purchase a backbone for its computer network that provides the greatest throughput at the least monetary cost. These objectives are highly dependent on each other as increased cost brings increased throughput and vice-versa. The term *objective* is used to refer to the goal of the MOP to be achieved and *objective space* is used to refer to the coordinate space within which vectors resulting from the MOP evaluation are plotted [5].

There are three main evolutionary approaches taken to solve MOPs; aggregation approaches, population based non-Pareto approaches, and Pareto-based approaches [6]. In this paper, we use the latter two approaches. The non-Pareto based approach implemented in this paper is a lexicographic approach [6]. This approach involves the rank ordering of objectives based on the priority associated with each. Essentially, each of the fitness functions are applied sequentially to a given population member.

The other multiobjective approach used here is a Pareto-based approach that utilizes the concepts of Pareto Dominance in determining the set of solutions [6]. The concept of Pareto Optimality is integral to determining which members dominate each other. A way to determine if one solu-

tion is "better," or dominates another, is a necessity here as well as in all problems. Pareto concepts allow for the determination of a set of optimal solutions in MOPs. Although single-objective optimization problems may have a unique optimal solution, MOPs have a possibly uncountable set of solutions, which when evaluated produce vectors whose components represent trade-offs in decision space. One key Pareto concept, Pareto Dominance, is defined mathematically as [5]:

Definition 1 (Pareto Dominance for Minimization Problems): *A vector $\vec{u} = (u_1, \ldots, u_k)$ is said to dominate another vector $\vec{v} = (v_1, \ldots, v_k)$ (denoted by $\vec{u} \preceq \vec{v}$) if and only if u is partially less than v; i.e., $\forall i \in \{1, \ldots, k\},\ u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$.* □

Pareto optimal solutions are those solutions within the search space whose corresponding objective vector components cannot be all simultaneously improved. These solutions are also termed *non-inferior*, *admissible*, or *efficient* solutions, and their corresponding vectors are termed *non-dominated*; selecting a vector(s) from this vector set (the Pareto Front set) implicitly indicates acceptable Pareto optimal solutions (**genotypes**). These solutions may have no clearly apparent relationship besides their membership in the Pareto optimal set. It is simply the set of all solutions whose associated vectors are nondominated; it is stressed here that these solutions are classified as such based on their *phenotypical* expression. Their expression (the non-dominated vectors), when plotted in criterion (**phenotype**) space, is known as the *Pareto Front* [5, 7].

3 Related Work

The central model in CDIS is a Computational Immune System (CIS) or Artificial Immune System (AIS), which is modeled on the biological immune system [8, 9]. Algorithms based upon a CIS are not exact models of the biological immune system; instead, they are abstractions of the immune system ideas applicable to the problem being solved, and they are implemented in some computer setting. Only concepts required for the particular application are mapped into the CIS, and natural continuous processes must necessarily be digitized to work on a digital computer [10].

CDIS, which is based on the negative-selection model, uses the concepts of self and non-self. The definitions of self and non-self are important to the proper functioning of the CIS, but each particular application has its own method for defining these concepts [10]. Ideally, the training set representing self is designed to represent as nearly as possible the "normal" activity that takes place at the location protected by the CIS.

Previous work exists in developing artificial and computational immune systems, and several research thrusts are underway in the area of computer defense [11, 12, 13, 14]. These techniques have shown considerable promise in intrusion detection. Our work, however, was inspired mainly by the work of the research groups of Dasgupta [15, 16, 17], Forrest [18, 19, 20, 21], and Lamont [22, 23, 10, 24, 25].

Dasgupta and Gonzalez performed network intrusion detection and developed a GA-based, Classifier-based decision support tool for assisting in the response to an intrusion [16, 17].

Forrest, Hofmeyr, and others have done extensive work with designing and using CISs for computer security applications. They have built CISs for host-based ID and network-based ID. Their host-based IDS defines self as sequences of system calls made by privileged programs, so it detects abnormal, or non-self, system calls [20, 26]. Their network-based IDS, called LISYS, uses three features for defining self: the source IP address, the destination IP address, and the TCP port. Only TCP SYN packets, which signal the start of a connection, are monitored by this IDS. Connections that occur frequently are considered part of self [19]. Our research differs from this work in several ways. For instance, CDIS uses 28 features, which include the three used by LISYS. Also, CDIS examines all TCP, UDP, and ICMP packets, rather than just monitoring the TCP SYN packets.

Lamont and others developed a hierarchical, distributed system called the Computer Virus Immune System (CVIS), which is a CIS that detects viruses. It was designed to manage sets of antibodies as they move through their lifecycles, to include sharing good antibodies throughout the system, handling issues like costimulation, alarms, and reporting, and producing any applicable reactions to old, new, or unknown attacks [22]. The capability of detecting network-based intrusions was added [24, 27] and the system was renamed to CDIS.

4 Problem Description

This section discusses the intrusion detection problem. The research goal and details of the problem domain are explained.

4.1 Intrusion Detection Problem Statement

The goal of our research is to develop an innovative type of antibody for the Computer Defense Immune System (CDIS); specifically, the new method is intended to detect attacks that are modified versions of existing attacks or stealthy version of existing attacks. Stealthy attacks may

take place over a long period of time, cover a large number of targets, or originate from a number of different, coordinated sources. Because these attacks are designed to be stealthy, they are hard to detect using current intrusion detection systems [24]. Current IDSs can be tuned to detect some stealthy attacks, but the resulting false alarm, or false detection, rate usually increases to an unacceptable level. Thus, new methods for detecting these types of attacks are needed.

New attacks may be modifications of existing attacks [24], so an approach for an ID system is to use knowledge of existing attacks to develop generalized detectors. These generalized detectors might have the ability to detect unknown attacks that are based on existing attacks or that are similar to existing attacks. Developing such generalized detectors is one aspect of the Intrusion Detection (ID) problem. This approach maps to the Time Series Prediction problem [28], in which a sequence of symbols is input and the correct output symbol must be predicted based on the input symbols. In this mapping, the input symbols are a sequence of network packets, and the output symbols represent whether the sequence is assumed to be an attack or not.

4.2 Approach

A network Internet Protocol (IP) packet is made up of a number of fields, including routing information, packet function, status flags, and content. Table 1 summarizes some of the main IP and Transmission Control Protocol (TCP) fields[1] that were found to be useful in earlier ID work [25]. Although the packet content or payload is an important part of each packet, it is not used in this research for two reasons. The main reason is that the size of the search space increases immensely if this field is used; the second reason is that existing signature-based detectors can be used to examine the content field in an efficient manner.

Network traffic consists of a sequence of packets, and an attack is also a sequence of packets. The packet features and relationships between features of multiple packets can be used to determine if a particular sequence of packets is an attack or not. Thus, the ID problem for this research focuses on the features shown in Table 1 along with the packet relationships shown in Table 2 to decide whether a particular sequence of packets is an attack or not.

The previous discussion motivates a new method for building antibodies for the CDIS. We call this new process "vaccination," since it is inspired at a high level by vaccination in a human. "Vaccination" injects existing knowledge about an attack into the CDIS to develop antibodies which detect that attack plus generalized versions of it. Knowledge about the attack, specifically the relationships between packets in the attack, is used to develop an attack signature. Using packet relationships generalizes the signature because exact details such as the Source IP address become relationships. This generalized pattern is then "injected" into the EP process to create antibodies, each of which is a finite state transducer[2] (FST) that detects the generalized pattern.

Developing antibodies using "vaccination" provides the ability to define patterns of known attacks and variations or modifications of known attacks. This method might also detect new attacks that have similar packet relationships as do existing attacks. In addition, this method allows for distinguishing between attack sequences and non-attack sequences because the FST can be built to accept an attack sequence while rejecting a non-attack sequence.

The genotype, or internal representation, of a detector in this scheme is an FST, which represents some regular language or pattern. The phenotype, or outward expression, of the detector is a "Detect" or "Not Detect" signal, which corresponds to the FST rejecting or accepting the word, which represents the network packets that may constitute an attack. The fitness value of a particular FST is dependent on two factors: whether it detects an attack correctly and whether it does not detect a non-attack string as an attack. Because there are multiple factors involved, a multiobjective approach to solving this problem seems a natural fit; the particular multiobjective approaches used are discussed in the next section. These FST-based detectors are used as antibodies in the CDIS architecture.

Table 1: Packet Features [25]

Field Name	**Possible Values**
IP Fields	**(All packets)**
IP Ident. Number	0-65535
IP Time to live (TTL)	0-255
IP Flags	0-65535
IP Overall Packet Length	0-65535
IP Source Address	Valid IP address
IP Dest. Address	Valid IP address
TCP-Only Fields	**(TCP packets only)**
TCP Source Port	0-65535
TCP Dest. Port	0-65535
TCP Seq. Num	0-4294967295
TCP Ack Num	0-4294967295
Individual TCP Flags (*PCWR, Echo, Urgent, Ack, Push, Reset, Syn, Fin*)	Boolean

[1] Only TCP packets are examined in this effort; however, other IP sub-protocols could be examined in a similar manner since the nature of the algorithm does not specifically exclude any protocol.

[2] The term "finite state transducer" as used here is the same as a Mealy-type finite state machine.

Table 2: Packet Relationships

IP Relationships (for all Packets)	TCP-Only Relationships
Same-IP-Class-A-Src	Same-TCP-Port-Src
Same-IP-Class-B-Src	Same-TCP-Port-Dest
Same-IP-Class-C-Src	Same-TCP-Flags
Same-IP-Class-D-Src	Same-TCP-Seq-Num
Same-IP-Class-A-Dest	Same-TCP-Next-Seq-Num
Same-IP-Class-B-Dest	Same-TCP-Ack-Number
Same-IP-Class-C-Dest	Same-TCP-Size
Same-IP-Class-D-Dest	
Same-IP-Flags	
Same-Packet-Len	
Same-IP-ID-Num	
Same-IP-TTL	

Figure 1 shows a pedagogical example FST as generated by this approach. The symbols inside the brackets on the transitions are the input symbols that cause a transition; these symbols represent which of the relationships from Table 2 are present between two sequential packets. The symbol after the colon represents the output symbol: "Detect" or "Not Detect."

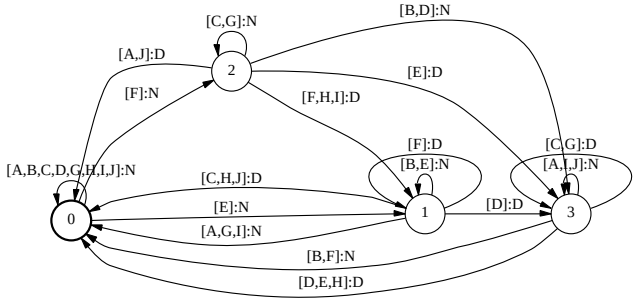


Figure 1: Example FST

5 Algorithm

Standard EP approaches have been modified to use representations such as finite state machines (FSMs) and FSTs, so appropriate reproductive operators have already been defined. EP has been repeatedly used to solve problems somewhat similar to the ID problem using only one objective. L. Fogel used EP to evolve FSMs that were capable of predicting a number in a series [29], while D. Fogel later modernized this work and added additional features [30]. Spears used EP to evolve FSMs that were capable of playing a game in which they defended network resources from an opponent [31]; this work implies that EP can be applied to network resource problems.

Given an input attack string, each FST produces some output string. A "detect" signal is generated when an FST outputs the sequence (N,N,...,N,D), where the length of the sequence is the same as the input string length. Thus, any output string that is not in the form of the "detect" signal is considered a non-attack.

Detectors are evolved through the EP process by selecting and reproducing the FSTs that best match the input attack string to the detect sequence. In addition, the algorithm uses some number of self-strings, or known non-attack strings, which the FSTs must not detect as attacks. The fitness functions discussed in subsection 5.1.2 are the embodiment of this process.

The original EP algorithm for this problem used only the single objective of detecting a given attack string, while the second version added the ability to use a single self-string along with the attack string in a lexicographic multiobjective fashion [32]. The current implementation provides the ability to use an arbitrary number of self-strings along with the attack string in either a lexicographic or Pareto-based multiobjective optimization. The selection mechanisms used for the current version are described in subsection 5.1.3.

The algorithm pseudocode is shown in Figure 2.

initialize population P of size n

evaluate all population members $p \in P$
with respect to all fitness functions

while Current Generation $<$ Max Generations **do**
mutate: For each $p \in P$,
Conduct Mutation Step 1 with probability 1.0
Conduct Mutation Step 2 with probability P_{Mut_2}
Conduct Mutation Step 3 with probability P_{Mut_3}

evaluate: all population members $c \in$ Child Population C
with respect to all fitness functions

select: Next parent population P using the appropriate
Tournament Selection routine
(Lexicographic or Pareto-based)
od

Figure 2: Pseudocode for Algorithm

5.1 Evolutionary Operators for the Algorithm

Since the algorithm for the current project was inspired by Fogel's work [30], it is similar to that EP algorithm, with differences such as the selection mechanism and the addition of the ability to use multiple objectives. The particular evolutionary operators used are discussed in the following subsections.

5.1.1 Reproduction

Mutation is the only reproductive operator used in this algorithm. It can take one of the following five forms: change the output symbol, change a state transition, add a state, delete a state, or change the start state [30]. Some internal bookkeeping is required so that a state is not deleted from an FST with only one state or a state is not added to an FST that already has the maximum number of allowed states [30].

One of these five mutations is performed on each population element during every generation; the specific mutation is chosen uniformly. A second or third mutation for each population element is also allowed; the user sets the probability of occurrence for each of these other mutations.

Recombination is not used in the algorithm, since mutation is historically the primary reproductive operator used for EP. However, future versions might incorporate recombination of some sort to further explore the search space. Since the search landscape has not been characterized for this problem, experimentation is required to determine what reproduction operators work well.

5.1.2 Fitness Functions

The first fitness function measures the percentage of FST output symbols that match the sequence (N,N,...,N,D), given the input attack string. This is the only fitness function used in the single-objective version of the algorithm. This fitness function is similar to that of the Boolean satisfiability problem [33]. The input string causes a sequence of output symbols to be generated by each FST; these symbols are then compared to the expected or desired output string of (N,N,...,N,D) to give a fitness value of absolute error in the generated output versus the expected output.

The other fitness functions, used in the multiobjective versions of the algorithm, measure the percentage of FST output symbols that do not match the sequence (N,N,...,N,D), given the particular "self" or non-attack string. Note that the higher the fitness, the fewer symbols match the "detect" sequence, but any fitness value above zero indicates the FST did not detect the "self" input string as an attack.

5.1.3 Selection

Standard tournament selection with replacement is employed in all versions of the algorithm. Any number of competitors is allowed in the tournament, so the selection pressure can be adjusted as desired.

In the single-objective version of the algorithm, only the first fitness function is calculated and used for the tournament process. Similarly, the lexicographic multiobjective version of the algorithm only uses the first fitness value for the tournament selection procedure. However, when a FST has a fitness of 1.0, which means it detects the attack correctly, the other fitness values are calculated to determine if the FST detects any of the "self" strings as an attack. If any of the strings are detected as an attack, then the FST receives a user-specified penalty on its first fitness score and then continues into the evolutionary programming process with the penalty. On the other hand, the Pareto-based multiobjective version of the algorithm uses all fitness values for every tournament.

6 Experimental Approach

The testing performed on the algorithm is designed to determine whether the two multiobjective EP algorithms are capable of finding solutions, or good detectors, over a range of input attack strings. In this case, a good detector is one which detects the input string developed from a sequence of attack packets and generates a "Detect" signal. Thus, the outputs of the algorithms are the time required to find a good solution and the number of generations needed to find that solution. The goal of the testing is to examine the efficiency of the algorithm rather than the solution effectiveness. Examining solution effectiveness is work that is currently on-going.

The algorithms are tested using two different test sets. The first test set consists of five representative input strings, or benchmark attack packets. These representative strings are generated randomly, using any of the possible relationships from Table 2. Multiple input string lengths, including strings much larger than those expected in the ID domain, are used in these tests. The second set of tests uses a real-world scan generated by the Queso tool. Each test run also uses five self strings that are generated by randomly changing two positions of the input attack string. The number of positions to change was chosen to keep the self strings "close" to the original attack string in terms of a distance metric. Thus, each test uses one attack string and five self strings, for a total of six fitness functions to be maximized.

6.1 Test Setup and Parameters

This testing is performed on a dedicated Dell Latitude Pentium III 1 GHz computer with 512 MB RAM and the Windows 2000 Professional Edition operating system. The algorithm is implemented in Java 1.3.1_01 and is compiled to a native Windows executable using the JOVE 2.0 compiler. The random number generator is Sean Luke's Java implementation of the Mersenne Twister algorithm, which has a longer period and is faster than the standard Java random number generator [34].

All of the test runs use the same parameters, other than the population size and number of generations allowed. These

tests allow a maximum of 15 states for any FSM, and the probability of a second mutation step is 1.0 and a probability of a third mutation step of 0.5. These parameters were chosen based on previous internal testing to show the feasibility of using the multiobjective approaches. For the first and second test sets, each experiment was repeated with population sizes of 100, 250, and 500. For the third test set, which uses a real attack, all of the tests used a population size of 100. All of the tests used ten replications for each test run, where each replication is allowed to execute until a solution is found.

6.2 Test Results and Analysis

Table 3 summarizes the average time and number of generations needed to find a solution FST for the first set of tests; the name of each test shows the type of algorithm used. The tests labeled with a "LX" use the lexicographic multiobjective approach, while the tests marked "PF" use the the Pareto-based method. Figures 3 through 7 show boxplots which present a graphical view of the primary test results for the number of generations and time required to find a solution and provide some appreciation for the distribution of the results. The boxplot provides a method for graphically depicting the distribution of a dataset and comparing multiple datasets. The box contains the middle 50% or the interquartile range (IQR) of the distribution. The unmarked line inside the box represents the median, while the line with square symbols represents the average. The lines extending from the top and bottom of the box contain any points within 1.5*IQR; any points outside of this range are shown as outliers [35].

As can be seen in all of the figures, there is a large variance in the time to execute and the number of generations required to generate a solution for the LX and PF tests. This is expected as the process is based on random events, but the variance and averages in the PF testing are significantly larger than those noticed in the LX tests. In addition, each of the Pareto-based tests has one outlier or large value that skews the average. For example, Figure 6 shows that the PF-4 test had an outlier that took 1443 generations to find a solution, while the median value for test PF-4 was only 72 generations.

The box plots also illustrate the fact that the time and generations required for each set of tests seem to be related. This observation is expected, since the time to execute one generation is fairly constant during the course of a test run.

Additionally, we note that the LX tests always converge to a solution in less time than the PF tests. At an initial glance this would lead us to state a preference for using the LX approach, but there is a need to further analyze the overall quality of the solutions found to determine which method performs statistically better or if they are equivalent in terms of solution quality. Solution quality, in this case, refers to the false alarm rate, which measures how often the solution incorrectly detects some benchmark set of non-attacks as attacks. The false alarm rate plays a key factor in determining the usefulness of an intrusion detection system, so it is important to test. Our conjecture is that the quicker execution time of the LX tests leads to a larger false alarm rate, or lower solution quality; solution quality and this conjecture will be tested in future work.

Table 3: Summary of Test Results

Test Name	*Avg Time (s)*	*Time Std Dev*	*Avg Gen*	*Gen Std Dev*
LX-1	22.49	12.50	41.10	53.73
PF-1	204.58	362.38	819.00	1537.29
LX-2	33.44	29.88	86.70	129.08
PF-2	187.91	293.72	740.20	1246.76
LX-3	29.37	19.27	71.30	83.09
PF-3	207.17	227.73	832.80	972.08
LX-4	32.51	27.08	81.50	119.47
PF-4	71.30	102.50	246.80	435.51
LX-5	32.68	17.35	80.00	74.71
PF-5	68.04	69.34	224.50	288.54

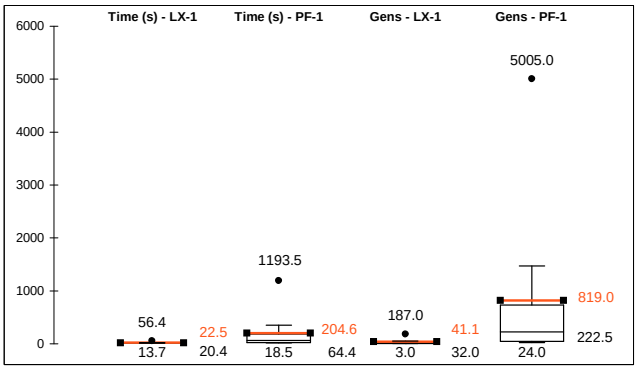


Figure 3: Comparison of Lex and Pareto-based for Test 1

The results of the second test set, which use the real-world Queso scan, are shown in Table 4. Figures 8 and 9 are boxplots which compare the generations and time required for the tests. The results for these tests follow the pattern of the first test set, in that the Lexicographic tests ran for fewer generations and less time, in general, before finding a solution. For example, the data in the table shows that the lexicographic test runs ran an order of magnitude faster with an order of magnitude less variance than the Pareto-based tests.

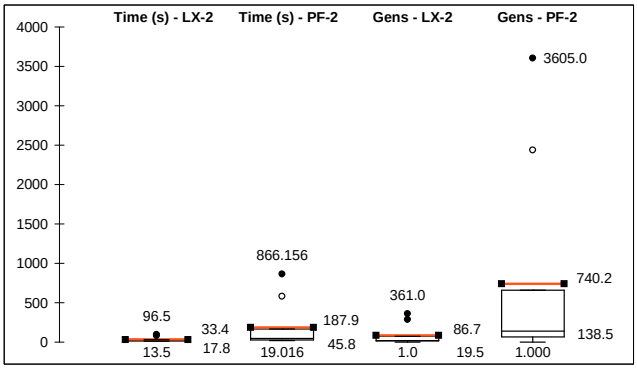


Figure 4: Comparison of Lex and Pareto-based for Test 2

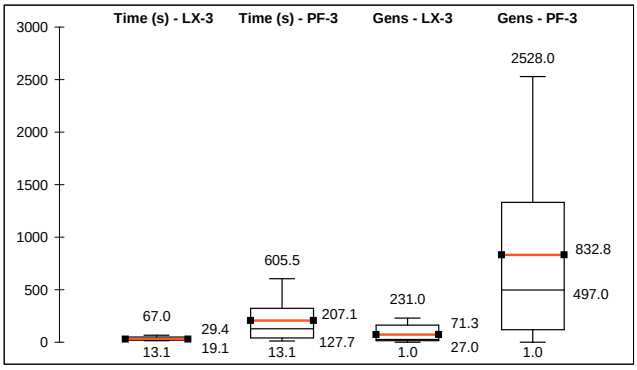


Figure 5: Comparison of Lex and Pareto-based for Test 3

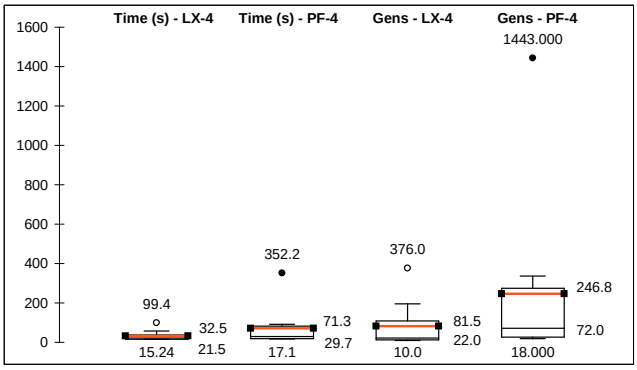


Figure 6: Comparison of Lex and Pareto-based for Test 4

Table 4: Summary of Test Results for Test Set 2 (Queso)

Test Type	*Avg Time (s)*	*Time Std Dev*	*Avg Gen*	*Gen Std Dev*
LX	296.1	224.1	338.4	285.6
PF	5441.3	6962.7	6750.0	8692.7

7 Conclusions

This research presents another step in detecting computer network intrusions through the use of a new type on anti-

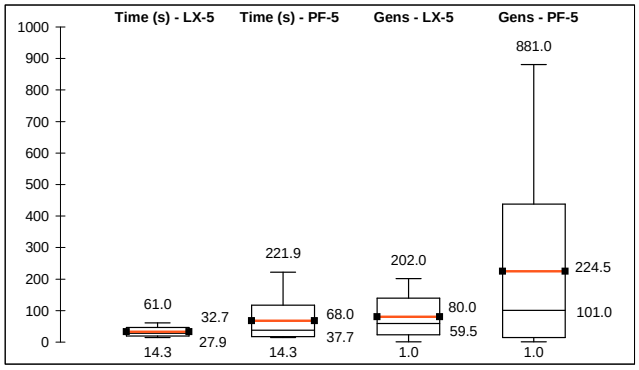


Figure 7: Comparison of Lex and Pareto-based for Test 5

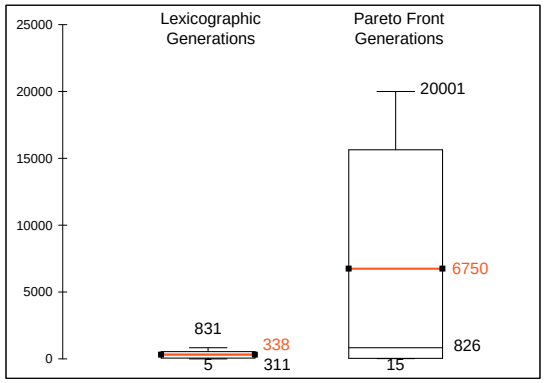


Figure 8: Comparison of Generations for Lexicographic and Pareto-based Queso Scan Test

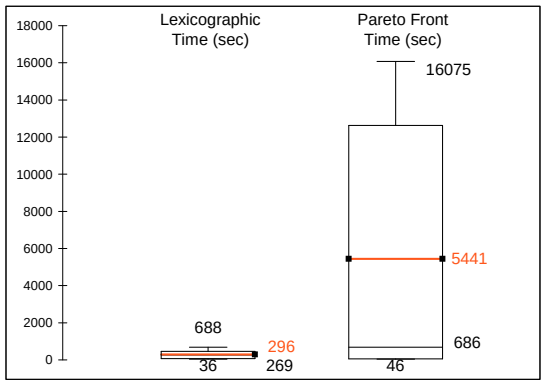


Figure 9: Comparison of Time for Lexicographic and Pareto-based Queso Scan Test

body for the Computer Defense Immune System. The antibodies are created through "vaccination" using knowledge about an attack in a multiobjective evolutionary programming algorithm.

The testing shows that the evolutionary programming technique generates finite state transducers, or Mealy-type finite state machines, capable of matching or detecting an

input attack string, for the limited string sizes tested. The lexicographic approach allows the use of the attack and "self" strings while performing significantly faster than the Pareto-based approach; however, further testing is required to determine the solution quality. Solution quality is determined by the false detection rate and the missed detection rate, but determining exactly how to measure solution quality in a fair and consistent manner is an on-going research question in the ID community. We are attempting to develop an appropriate test bed with "real-world" data so that the solution quality of the antibodies generated by the "vaccination" process can be tested.

Acknowledgments

This work is sponsored in part by the Air Force Office of Scientific Research.

References

[1] S. Axelsson, "Intrusion detection systems: A survey and taxonomy," Tech. Rep. 99-15, Department of Computer Engineering, Chalmers University, 2000.

[2] Thomas Bäck, D. B. Fogel, and Z. Michalewicz, Eds., *Evolutionary Computation 1: Basic Algorithms and Operators*, Institute of Physics, Bristol (UK), 2000.

[3] David A. Van Veldhuizen and Gary B. Lamont, "Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art," *Evolutionary Computation*, vol. 8, no. 2, pp. 125–147, 2000.

[4] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, 233 Spring St., New York, NY 10013, 2002.

[5] David A. Van Veldhuizen, *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*, Ph.D. thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.

[6] C. M. Fonseca and P. J. Fleming, "Multiobjective Optimization," in *Evolutionary Computation 2 Advanced Algorithms and Operators*, Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, Eds., vol. 2, pp. 25–37. Institute of Physics Publishing, Bristol (UK), 2000.

[7] Jesse B. Zydallis, David A. Van Veldhuizen, and Gary B. Lamont, "A Statistical Comparison of Multiobjective Evolutionary Algorithms Including the MOMGA–II," in *First International Conference on Evolutionary Multi-Criterion Optimization*, Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, Eds., pp. 226–240. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[8] Dipankar Dasgupta, Ed., *Artificial Immune Systems and Their Applications*, Springer-Verlag, Berlin, 1998.

[9] D. Dasgupta, N. Majumdar, and F. Nina, "Artificial immune systems: A bibliography," Tech. Rep. CS-01-002, Version 2.0, Computer Science Division, University of Memphis, 2001.

[10] Robert E. Marmelstein, David A. Van Veldhuizen, Paul K. Harmer, and Gary B. Lamont, "Modeling & Analysis of Computer Immune Systems using Evolutionary Algorithms, Revision 2," White Paper, December 1999, Air Force Institute of Technology, Wright-Patterson AFB, OH.

[11] Jungwon Kim and Peter Bentley, "The Artificial Immune Model for Network Intrusion Detection," in *7th European Conference on Intelligent Techniques and Soft Computing (EUFIT'99)*, Aachen, Germany, 1999.

[12] Jungwon Kim and Peter Bentley, "Towards an Artificial Immune System for Network Intrusion Detection: An Investigation of Clonal Selection with a Negative Selection Operator," in *Congress on Evolutionary Computation (CEC-2001)*, Seoul, Korea, 2001, pp. 1244–1252.

[13] Jungwon Kim and Peter Bentley, "An Evaluation of Negative Selection in an Artificial Immune System for Network Intrusion Detection," in *Genetic and Evolutionary Computation Conference 2001 (GECCO-2001)*, San Francisco, CA, 2001, pp. 1330 – 1337.

[14] Jungwon Kim and Peter Bentley, "Towards an Artificial Immune System for Network Intrusion Detection: An Investigation of Dynamic Clonal Selection," in *Proceedings of the 2002 Congress on Evolutionary Computation*, Honolulu, 2002, pp. 1015–1020, IEEE Press.

[15] Dipankar Dasgupta, "Immunity-Based Intrusion Detection Systems: A General Framework," in *Proceedings of the 22nd National Information Systems Security Conference (NISSC)*, 1999.

[16] Dipankar Dasgupta and Fabio A. Gonzalez, "A new approach to intrusion detection," University of Memphis, C. S. Technical Report No. CS-01-011, May 2001.

[17] Dipankar Dasgupta and Fabio A. Gonzalez, "An Intelligent Decision Support System for Intrusion Detection and Response," in *Lecture Notes in Computer Science, Proceedings of the International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security (MMM-ACNS)*, St. Petersburg, Russia, 2001, Springer-Verlag.

[18] Stephanie Forrest and Steven A. Hofmeyr, "Immunology as Information Processing," in *Design Principles for the Immune Systems and Other Distributed Autonomous System*, pp. 361–388. Oxford University Press, 2001, Available electronically at URL `ftp://ftp.cs.unm.edu/pub/forrest/iaip.ps`.

[19] Steven Hofmeyr and Stephanie Forrest, "Architecture for an Artificial Immune System," *Evolutionary Computation*, vol. 7(1), pp. 1289–1296, 1999.

[20] Steven Hofmeyr, Stephanie Forrest, and A. Somayaji, "Intrusion Detection using a Sequence of System Calls," *Journal of Computer Security*, vol. 6, pp. 151–180, 1998.

[21] A. Somayaji, S. Hofmeyr, and S. Forrest, "Principles of a Computer Immune System," in *Proceedings of the New Security Paradigms Workshop (NSPW-97)*, Langdale, United Kingdom, 1997, pp. 75–82, Association for Computing Machinery.

[22] Paul Harmer, "A Distributed Agent Architecture for a Computer Virus Immune System," M.S. thesis, AFIT/GCE/ENG/00M-02, Graduate School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, March 2000.

[23] Paul Harmer, Paul Williams, Gregg Gunsch, and Gary Lamont, "A Distributed Agent Based Architecture for Computer Security Applications," *To Appear in IEEE Transactions On Evolutionary Computation, Special Issue on Artificial Immune Systems*, 2001.

[24] Paul D. Williams, "Warthog: Towards a Computer Immune System for Detecting "Low and Slow" Information System Attacks," M.S. thesis, AFIT/GCS/ENG/01M-15, Graduate School of Engineering and Management, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, March 2001.

[25] Paul Williams, Kevin Anchor, John Bebo, Gregg Gunsch, and Gary Lamont, "CDIS: Towards a Computer Immune System for Detecting Network Intrusions," in *Proceedings of the 4th International Symposium, Recent Advances in Intrusion Detection 2001*, Berlin, 2001, pp. 117–133, Springer-Verlag.

[26] Justin Balthrop, Stephanie Forrest, and Matthew Glickman, "Revisting LISYS: Parameters and Normal Behavior," in *Proceedings of the 2002 Congress on Evolutionary Computation*, Piscataway, NJ, 2002, pp. 1045–1050, IEEE Press.

[27] Kevin Anchor, Paul Williams, Gregg Gunsch, and Gary Lamont, "The Computer Defense Immune System: Current and Future Research in Intrusion Detection," in *Proceedings of the 2002 Congress on Evolutionary Computation*, Honolulu, 2002, pp. 1027–1032, IEEE Press.

[28] David B. Fogel, *Evolutionary Computation: Principles and Practice for Signal Processing*, SPIE Press, P.O. Box 10, Bellingham WA 98227, 2000.

[29] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Intelligence*, John Wiley, NY, 1966.

[30] David B. Fogel and Kumar Chellapilla, "Revisiting Evolutionary Programming," in *SPIE Aerosense98, Applications and Science of Computational Intelligence*, S.K. Rogers, D.B. Fogel, J.C. Bezdek, and B. Bosacchi, Eds., Orlando, FL, 1998, pp. 2–11.

[31] William Spears and Diana Gordon, "Evolving Finite-State Machine Strategies for Protecting Resources," in *Proceedings of the International Symposium on Methodologies for Intelligent Systems 2000*. 2000, ACM Special Interest Group on Artificial Intelligence.

[32] Kevin Anchor, Gary Lamont, and Gregg Gunsch, "An Evolutionary Programming Approach for Detecting Novel Computer Network Attacks," in *Proceedings of the 2002 Congress on Evolutionary Computation*, Honolulu, 2002, pp. 1618–1623, IEEE Press.

[33] Z. Michalewicz and D. Fogel, *How to Solve It: Modern Heuristics*, Springer-Verlag, Berlin, 2000.

[34] M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator," *ACM Transactions on Modeling and Computer Simulation*, vol. 8(1), pp. 3–30, 1998.

[35] Stephen Vardeman and Marcus Jobe, *Statistical Qualtiy Assurance Methods for Engineers*, John Wiley and Sons, Inc., New York, 1999.

AISIMAM – An Artificial Immune System Based Intelligent Multi Agent Model and its Application to a Mine Detection Problem

Srividhya Sathyanath
Dept of Electrical Engineering,
Rochester Institute of Technology,
sxs4446@rit.edu

Ferat Sahin
Dept of Electrical Engineering, RIT
79 Lomb Memorial Drive,
Rochester, NY 14623
feseee@rit.edu

Abstract

Artificial Immune System (AIS) is a novel evolutionary paradigm inspired by the biological aspects of the immune system. The human immune system has motivated scientists and engineers for finding powerful information processing algorithms that has solved complex engineering tasks. This paper discusses two concepts. (a) The behavioral management of artificial intelligence (AI) namely the *intelligent multi agent systems*, (b) The evolutionary computation called the *artificial immune system* that imitates the biological theory called the immune system. The outcome of this research is an **A**rtificial **I**mmune **S**ystem based **I**ntelligent **M**ulti **A**gent **M**odel named ***AISIMAM*** that solves agent-based applications. The model is applied to a mine detection and diffusion problem and the results prove that ***AISIMAM*** has solved the problem successfully.

1 Introduction

The study of biological systems is of interest to scientists and engineers as they turn out to be a source of rich theories. They are useful in constructing novel computer algorithms to solve complex engineering problems. *Genetic algorithms* derived from the principles of genetics, *Neural Networks* derived from brain - nervous systems or neurology (Dasgupta & Attoh-Okine, 1997) and *cellular engineering* based on cell biology are some of the biologically motivated evolutionary algorithms that perform information processing tasks. *Immunology* as a study of the immune system (Elgert, 1996) inspired the evolution of *artificial immune system*, which is an area of vast research over the last few years. Artificial immune system imitates the natural immune system that has sophisticated methodologies and capabilities to build computational algorithms that solves engineering problems efficiently. The main goal of the human immune system is to protect the internal components of the human body by fighting against the foreign elements such as the fungi, virus and bacteria (Timmis et al., 1999). It is interesting to observe that the process of recognition, identification and post processing involve several mechanisms such as the pattern recognition, learning, communication, adaptation, self-organization, memory and distributed control by which the body attains immunity (Dasgupta, 1999).

AIS has made significant contributions to machine intelligence. Applications of AIS are not limited to *optimization, robotics, neural network approaches, data mining* and *image classification* (Hajela & Yoo 1999; Ishiguro et al., 1997; Hoffmann 1986; Hunt & Fellows 1996; Sathyanath & Sahin, 2001).

In this paper, we concentrate on *Multi Agent Systems (MAS)* and their characteristics. Multi agents are population of *agents*, (i.e.), more than one agent reacts to the change in environment to accomplish the task (Huhns & Singh, 1998). Multi agent systems are based on behavior management of several independent agents (M. Wooldridge, 1999).

The objective of the authors was to develop a biological based intelligent multi agent architecture. Multi agent systems have some features in common with the immune system and provide scope for applying immune system methodologies. Therefore, we have applied artificial immune system to multi agent systems for the computational intelligence of agents. The outcome of the research is a generic ***A**rtificial **I**mmune **S**ystem* based ***I**ntelligent **M**ulti **A**gent **M**odel* named ***AISIMAM***. The model draws an analogy between the immune system and agent methodologies. It applies the immune system principles to the agents to perform a global goal in a distributed manner. *AISIMAM* is applied to mine detection and diffusion problem, a specific application experimented to prove the model. This paper shows that *AISIMAM* solves the mine detection application successfully.

The organization of this paper is as follows. Section 2 presents a brief introduction to the immune system. Section 3 discusses agent definitions, characteristics of multi agents in problem solving. Section 4 focuses on *AISIMAM* with the mathematical derivations and explantions. Section 5 explains the need for the mathematical representation and Section 6 demonstrates the application of *AISIMAM* to a mine detection and diffusion problem. In Section 7 we state the new aspect of this research and in Section 8 we state the scope for

future work. Section 9 summarizes the conclusion derived out of this research work.

2 The Human Immune System

The natural immune system is a very complex system with several mechanisms for defense against infectious agents entering our system. The external components to the immune system are *antigens* or called the *non-self cells*, as they are foreign substances to the body. The basic components of the immune system are the white blood cells, called *self-cells* or *lymphocytes* in immunological terms. These specialized cells are classified into two types namely the *B lymphocytes* and *T lymphocytes*.

- *B-lymphocytes* are the cells produced by the bone marrows
- *T* cells develop in bone marrow and mature in *thymus*

The major responsibility of the *B* cells is the secretion of the receptors called the *antibodies* (*Ab*) as a response to the *antigens* that enter the body (*Ag*) (Hajela & Yoo, 1999). The role of these receptors on the surface of the *B* cell is to recognize and bind the antigen. These receptors are called *idiotopes* and *paratopes*. Antigens also have receptors called *epitopes*. The *B* cells generate antibodies of *complementary match* that recognizes and binds the antigen (Castro & Von Zuben, 1999). Complementary match means the generation of an opposite shape or structure that fits well with the antigenic epitope to recognize the antigen. The receptors of the *B* cell change their shape according to the shape of the epitope (Timmis et al., 1999). Figure 1 shows the *B* cell, *B* cell receptors and the epitopes of the antigen.

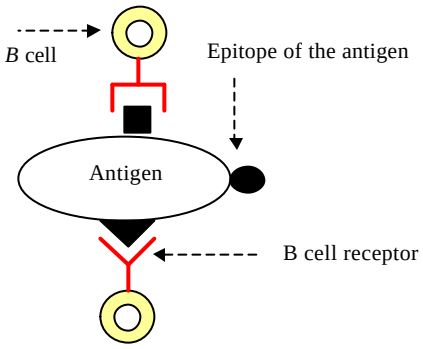


Figure 1: *B* cells, *B* cell receptors, antigen, and epitopes.

2.1 Properties of the Human Immune System

This section briefly discusses some of the properties of the immune system by which the human body attains immunity. The main function of the immune system is to kill the antigen. It is interesting to note that this common goal of the system is handled by the individual components of the immune system in a distributed fashion. At the same time they also have remarkable properties with which they work collectively to perform the task.

The immune system possesses the following properties.

- *Positive and negative selection* is a process of discrimination of self/non-self cells that prevents autoimmuno diseases. This process filters out the cells that would work against the self-cells and only the cells that would not bind the self-cells circulate to fight against the antigens (Dasgupta, 1999).
- *Clonal selection and expansion* is a process of selection of useful cells that recognize the antigen and reproduce those cells. This process of *cloning* multiplies the useful cells that are capable of recognizing the antigens. Therefore, the *B* cells that contain the specific receptor that match a particular antigen are also multiplied. In this process, the clones suffer *hypermutation* that alters the shape of the receptor also called *receptor editing*, thus increasing the affinity between the clone and the specific antigen (Burnet, 1978; Dasgupta, 1999).
- *Immune memory* is a result of clonal expansion. Some of the cloned cells differentiate into *memory cells* and the rest of the clones become *plasma cells*. *B* cells remember the shape of the antigen that they have fought and recollect when they see the same antigen again. This process defined as *secondary response*, is a feedback of the past event for a current input. This process helps the system to learn and is called as *reinforcement learning*. Plasma cells produce cells with higher affinities (Castro & Von Zuben, 1999).
- *Jerne's idiotropic network* deals with the interaction of antibodies. Jerne's network is a network of *B* cells that communicate the shape of the antigenic epitope amongst them through idiotopes and paratopes. This also transforms the receptors according to the antigenic pattern. This shape transformation is an important role of information transfer and communication between the *B* cells (Jerne, 1984).

Figure. 2 show the overall functioning of the immune system. The immune system recognizes the antigens and the antigenic patterns are identified. On identification of an antigenic pattern, the *B* cells communicate the information in parallel to each other by means of paratopes and idiotopes in the network. Paratopes match with the epitopes of the antigen to recognize the antigen. Paratopes also change their shape to strengthen the bond between the epitope and the paratope. However, the binding stays only for a short time called the *tolerization period* (Hofmeyer, 2000) within which a number of receptors should bind the antigen. When this process of binding within a short period happens, the *B* cells gets activated and performs a set of actions to kill the antigen (Hofmeyer, 2000). On activation, every *B* cell responds by changing the shape of the receptor according to the antigenic epitope. *B* cells that have higher affinity towards the antigen are the ones that recognize the antigen. The useful cells undergo multiplication by clonal

expansion and produce high affinity cells or clones. Since the antigen has multiple epitopes and the *B* cells are *monospecific* (Castro & Von Zuben, 1999) with a single type of receptor, *B* cells work together to kill the antigen through immune network. Part of the clones differentiate into plasma cells that create higher affinity cells and the rest turn out to be memory cells that remember the antigen that was destroyed. Thus the human system attains immunity against the antigens.

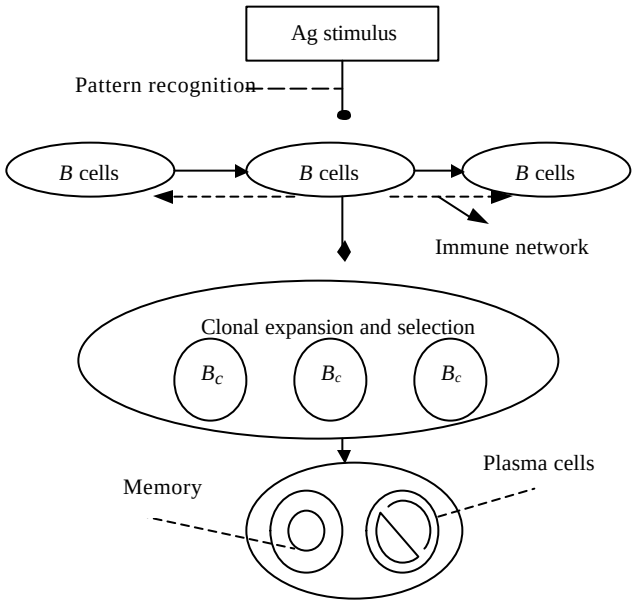


Figure 2: Representation of the human immune system.

3 Multi Agent Systems

Multi agent systems (*MAS*) deal with the behavior management in collection of several independent entities, or *agents* (Wooldridge, 1999). There are several definitions for agents. We have chosen two definitions of agents.

- Nwana and Ndumu defined an agent as "a component of software and/or hardware which is capable of acting in order to accomplish tasks on behalf of its user" (Nwana & Ndumu, 1997).
- Agents that operate robustly in rapidly changing, unpredictable, or open environments, and where there is a significant possibility that actions can fail are known as *intelligent agents* or sometimes called *autonomous agents* (Bond & Gasser, 1998).

Agents can exist alone or in a society of agents called *multi agents* (*MAS*). Multi agents are a population of agents, that is, more than one agent can change the environment to accomplish the task. They are distributed computational systems (Cho & Tae-Lim, 2001) in which each agent in MAS has a list of individual goals or tasks that it will perform. At the same time, MAS has global goals that all the agents will strive to achieve where the individual efforts of each member agent are put together toward reaching the MAS's global goals (Huhns & Singh, 1998). The advantage of the MAS is that the limitations of the individual capabilities of the agents are eliminated (Abul et al., 2000). Agents with a fixed goal learn how to change the environment to achieve the end goal. This process is called *reinforcement learning* in agents. In order to achieve an independent and global problem solving, the agents behave according to its defined characteristics. Some of the characteristics of agents that define their behavior are autonomy, friendliness, reasoning, learning, communication and coordination mechanisms. Similarly, there are different environments according to which the agents perform the goals. The multi agent environment is usually open, decentralized, and contains autonomous agents (Huhns & Stephens, 1999). In summary, agents are entities with well-set goals, actions and knowledge in an environment that senses, communicates, coordinates, learns and makes decisions according to the environment (Cho & Tae-Lim, 2001). The following section briefly describes some of the characteristics of the agents and different kinds of environment (Mohammed, 2000).

3.1 Characteristics of the Agents and the Environment

The characteristics of the agents are as follows.

1. Autonomy in agents is a measure of self-sufficiency. The agents that operate on their own are *independent* agents, and if they are restricted by external influences then they are called *controlled* agents.
2. Sociability is a behavioral measure of an agent to think about itself or about others. An *altruistic* agent acts regardful of others benefits, and is unselfish. In contrast, an *egoistic* agent acts with excessive thoughts of self and is self-loving.
3. Agents could be friendly and be *cooperative* or *compete* with each other.
4. Agents are classified into *reactive* and *deliberative* according to their level of *cognition*. The former ones sense and react in a timely manner for an environmental change and the latter ones reason out before making actions.
5. Mobility determines if the agents are *stationary* or *itinerant*. Stationary agents do not move and itinerant agents are mobile. Other characteristics of the agents that deal with the agent's adaptability, rationality and locality can be referred to the literature (Mohammed, 2000).

An agent may have a problem in deciding which of its actions it should perform in order to best satisfy its design objectives. The complexity of the decision making process can be affected by a number of different environmental properties. The following are various environments stated by Russell and Norvig. (Russell &

Norvig, 1995; Mohammed, 2000).
An *accessible* environment is one in which the agent can obtain complete, accurate, up to date information about the environment's state. The more accessible an environment is, the simpler it is to build agents to operate on it. Complex environments like the physical world are defined as *inaccessible* environments.

There are also other kinds of environments. *Deterministic* environment and *non-deterministic* environment deals with the certainty of agent's action. *Episodic* and *non-episodic* environment deals with the performance of agent's in discrete episodes without any links or linked actions with the past and current data respectively.

4 *AISIMAM* - Artificial Immune System Based Intelligent Multi Agent Model

The backbone of *AISIMAM* involves imitating the human immune system in terms of features and functions in multi agent systems. The motivation for this research comes from the fact that artificial immune system has found solutions for several applications. In the same context agent based solutions have also been developed in different application domains (Cho & Tae-Lim 2001, Abul et al., 2000). The reason for developing the *AISIMAM* is due to the similarities observed between the immune system architecture and the architecture of the agents. The distinct similarities between the agents and the immune system are

- Both are distributed or decentralized systems
- Both have multiple autonomous entities
- Both have individual and global goals
- Both systems learn from their experience
- Both are adaptable
- Both sense the changes in the environment and act accordingly
- Both systems communicate and coordinate
- Both possess knowledge with which they make intelligent decisions.

Therefore, immune system based multi agent architecture is derivable. The following section describes the multi agent systems with necessary comparisons and explanations.

4.1 Comparison of AIS and Multi Agent System Parameters

The model defines the non-self cells (antigens) and self-cells (*B* & *T* cells) as two agents with different characteristics and goals. Therefore, the two types of agents in *AISIMAM* are

- Antigens are modeled as non-self agents (*NAGs*) and
- Lymphocytes or self-cells corresponds to self-agents (*SAGs*)

We define the *environment* to be a matrix in which both the *NAGs* and the *SAGs* operate. The environment can be any one of the types of environment explained in section 3.1 depending on the application. We assume that there is an information vector for each non-self agent. This could represent a disturbance in a process, malfunction or a virus in a computer network depending on the application. The information vectors correspond to the epitopes of the antigen. Similarly, each self-agent has an information vector that defines the self-goals. The information vectors correspond to the receptors of the lymphocytes. The information vector can contain a single datum or multiple data. For example, the information could be a location information, identification number, text information, or all of them depending upon the application. We consider this information to be the idiotopes and the paratopes. However, the model does not distinguish between the paratopes and idiotopes. Instead, the target will be to perform the end goal with the available information by each self-agent. The end goal could be destroying the non-self agent as the antigen is killed in the IS, or it can be to identify the best action sets of each self-agent to react to the non-self agent's action vector. This is however problem dependent.

The information vectors and the characteristics of the self and the non-self agents differ from each other. This is similar to the structures of the epitopes of the antigen and the paratopes of the lymphocytes. In other words, the agents perform individual actions or goals determined by the *action generator* function and the global goal is the coordinated actions of the individual *SAGs.* The individual action of the agent corresponds to the receptor shape change in a *B* cell and the coordinated actions correspond to a group of *B* cells killing the antigen.

The *SAGs* are assumed to have sensory capability to identify the *NAG* within a region called *sensory neighborhood*. They also possess the capability to communicate the *NAG* information to the other *SAGs* within a region called *communication neighborhood*. The model assumes that the communication neighborhood is greater than the sensory neighborhood. This is in comparison with the capability of the *B* cells to recognize the antigenic pattern within a particular neighborhood. In immune system, the communication circle is analogous to communication between *B* cells connected in the immune network (Jerne's Network). In other words, every *B* cell communicates the information to another *B* cell that is within the communication neighborhood in the immune network.

The agent model describes five stages of processing namely *Pattern recognition, Binding process, Activation process, Post activation process and Post processing.*

In pattern recognition, *SAGs* recognize the presence of the antigen by the *stimulation function* and identifies the *NAGs* by an *identifier function*. The model defines an *affinity function* that calculates an affinity value between the actions of the self and the non-self agents. This process is defined as *binding process*. In the immune system, the affinity is proportional to the binding between the *B* cell receptors and the epitopes. The affinity

calculation in the agents is similar to the affinity between the epitope of the antigen and the receptor of the antibody. However, the binding is not modeled separately in *AISIMAM*. For instance, the affinity function could be a distance metric such as the Euclidean distance.

In order to imitate the IS, in the activation process we choose the affinity values that are greater then a set *activation threshold*. Activation threshold will help the agents to find out the higher affinity actions called *mature actions* that are closer to the desired goal. Here, we define the *binding period* as the time taken by a number of agents to bind the *NAG*. The model defines this time as a sum of *recognition time* and *grouping time*. Recognition time is the time taken by every agent to recognize the *NAG* and is the same for every agent. The grouping time is the time taken by the other agents to react to the identified *NAG* and this time differs from agent to agent.

The post activation process involves cloning. Here, the agents are reproduced with the mature action. A part of these *cloned agents* differentiate into *memory agents* containing the matured action obtained as a result of a particular *NAG*. The rest of the clones become *plasma agents* that create higher affinity actions through the action generator function. Post processing involves the primary and secondary response of immune memory, which is also included in the model. Hypermutation in agents is the process of generating new actions exists conceptually. Once the end goal is reached, memory agents remember the actions performed to reach the goal.

All the self-agents work in an *agent network* similar to Jerne's network. The process of information transfer and communication between the agents is an analogy of the agent network to the immune network. The nature of the agent network is application dependent. Suppression in the agent network is determined by the *suppression function*. In immune system, even in the absence of the antigenic stimulus, the *B* cells perform suppression. In *AISIMAM*, in the absence of antigenic stimulus suppression is performed. The overall representation of the *AISIMAM* is shown in Figure 3.

4.2 *AISIMAM* - Operational Scheme and the Mathematical Representation

This section deals with the notations used in the model, followed by the definitions of the parameters, and the algorithm.

4.2.1 Parameter Definitions

In the model, we define the agents namely the self agents (*SAGs*) and represent them by S_i, where $i = 1, 2 \ldots N$ and the non-self agents (*NAGs*) as N_j where $j = 1,2 \ldots M$. We define the problem domain or the environment E by $E = S_i \cup N_j \quad \forall i, j$. For all $S_i \in E$, there exists an information vector of n elements given by $B^i = [b_1, b_2 \ldots b_n]$. For all $N_j \in E$, there exists an information vector of m elements given by $A^j = [a_1, a_2 \ldots a_m]$. Define T_a to be the activation threshold.

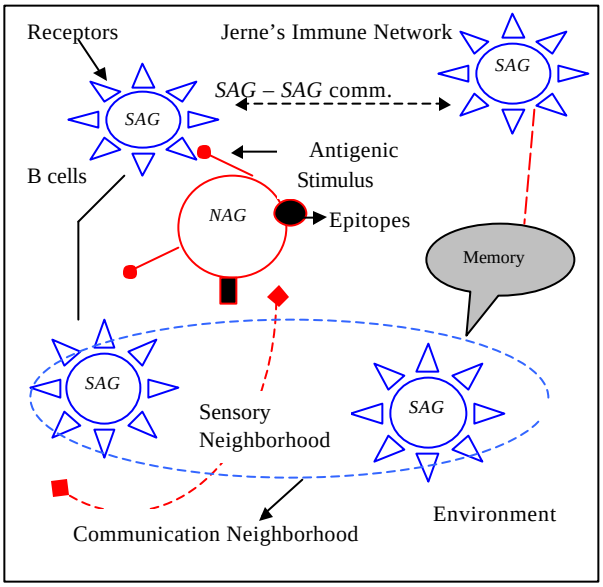


Figure 3: Representation of *AISIMAM* – An AIS based Intelligent Multi Agent Model

4.2.2 *AISIMAM* - Algorithm

Initialize all the parameters defined as above

For each S_i

- Calculate $M_{j,i} = f_1(A^j, B^i)$ where B^i is the information vector of S_i, and A^j is the information vector $\forall N_j$ in the sensory neighborhood N_s

$$f_1(A^j, B^i) = \begin{cases} 0 & \text{if no } A^j \\ \neq 0 & \text{if } \exists \text{an } A^j \text{ in } N_S \end{cases}$$

- **If** $(M_{j,i} \neq 0)$
  - The information about the *NAG* is transmitted to the other *SAGs* through the immune network
  - **For** each *NAG* N_j, within the N_s, the sensory circle where $j = 1, 2 \ldots e$, and $e \pounds M$

1. Pattern Recognition and Identification

Identify the *NAG* using the identifier function I that is given by

$$I_j = f_2(A_j)$$

Generate possible new actions $U^i_j \ldots\ldots U^i_k$ using action generator function that is a function of I_j

$$U^i_j = f_3(I_j) \text{ where } j = 1 \ldots . k$$

2. Binding Process

Find the affinity for all possible vectors U^i_j by the affinity function

$$Af_j^i = f_4(U^i{}_j), \forall j = 1...k$$

3. Activation Process

Choose mature actions whose affinity is greater than activation threshold T_a and store in the action set Y

$$Y = \left\{ U_j^i \mid Af_j^i > T_a \right\} \text{where } j = 1,2.....p$$

- The activation of the mature actions within the binding period t_b is given by

$$U_j^i = f_5(Y, t_b) * \left[u(t) - u(t - t_b) \right]$$

where $u(t)$ is the unit sep response

$$f_5(Y, t_b) = \begin{cases} 0 & \text{if no activation} \\ \neq 0 & \text{if } \exists \text{ activation} \end{cases}$$

If a best action needs to be chosen, the threshold should be chosen so high that $p = 1$.

4. Post activation processing - Cloning

If $\left(U_j^i \neq 0\right)$

In this case, **agents are reproduced** with mature action set Y in *SAGs.* S_i is cloned with mature action set Y to generate q *SAGs.*

S^c where $c = N+1, ..., N+q$

End If

5. Post processing - Memory

Choose s number of memory agents M_z^a from the cloned agents

If $\left(U_j^i \neq 0\right)$

$M_z^a = S^c$

where $z = N+1, \cdots, N+s$, where $s < q$

- **Memory Response**

The efficiency of the primary and secondary responses are given by

$$h_p = f_6[N_p, T_p]$$

$$h_s = f_7[N_s, T_s]$$

where $T_p >> T_s$ and $N_p << N_s$ and N_p and N_s are the number of actions required to kill the *NAG* in the primary response. T_p & T_s are the time taken for the primary and secondary responses respectively. The efficiency of the primary and secondary responses is h_p and h_s respectively.

- **Plasma Response**

Rest of the clones are defined as plasma agents S^z where $z = N+s+1,.....,N+q$. Here q-s agents are added into the system.

End If

End For

Else perform suppression by the suppression function

$P_{i,j} = f_8(B^i, B^j)$ where i, j are of S_i and S_j

End If

End For

5 Need for a Mathematical Representation

The goal of *AISIMAM* is to provide a mathematical representation for the operation of immune system. Several immune modeling such as the immune network model (Castro & Von Zuben, 2001), negative selection algorithm (Dasgupta), mathematical modeling of the clonal selection (Chowdary, 1999) and immune memory (Smith et al., 1996) agent based immune systems (Mori, Tsukiyama and Fukuda 1997, Dasgupta 1998) exist in the literature. *AISIMAM* differs from the other models in the context of mathematical functions defined for the entire process. In order to prove the usefulness of the representation, two applications namely bar code recognition and mine detection are compared.

In the case of barcode recognition, assume that the non-self agents N_j or antigens are the characters to be recognized. The *B* cells are the software agents S_i whose information vector contains the corresponding *ASCII* characters. Each agent has a defined group of characters. Environment *E* has the information about the recognized and the unrecognized characters. If the agent can recognize the character, recognition is achieved. Otherwise the agents can communicate through the environment to find if the unrecognized character falls into its category. The stimulus *M* is defined by the recognition of the start bit pattern of the barcode that defines the start of the recognition process. The identifier function *I* is a character recognition function. The affinity function *Af* can be defined as the matching function between the recognized character and the character in the agent's

information vector. Affinity threshold T_a can be set to 1 that chooses the best match. In this case cloning is not utilized. Thus the agents are not reproduced. In this application, sensory and communication neighborhood is zero, since the agents are not in a space.

In the case of mine detection application, non-self agents are the mines and the mobile robots are the self-agents. In this case, both are hardware agents. The sensory and communication neighborhoods are defined by the distance metric. The identifier function *I* becomes finding the mine by the identifier and the location of the mine. The affinity function *Af* is the Euclidean distance. Affinity threshold T_a can be set to a predefined value. Mine detection application is explained in detail in the following paragraphs.

As can be seen above, the model can be applied to different applications by changing the functions. Therefore, the generalized functions provide a global representation for several agent based applications.

6 Application of *AISIMAM* to a Mine Detection Problem

To experimentally verify the architecture, *AISIMAM* is applied to a specific problem. The problem implemented is mine detection and diffusion. The experiment is

simulated in MATLAB. The following section discusses the parameters of *AISIMAM* used for this specific application and the pseudo code for the problem.

6.1 Parameter Definitions

The following section briefly describes the characteristics of *NAGs*, *SAGs* and environment for mine detection.

6.1.1 *NAGs* and its characteristics

The antigen or the Nonself agent (*NAG*) is the mine. Define the area to be explored for detecting the mine. This defines the boundary of the environment for the agents to detect the mine. Mines are deployed in a uniform distribution within the environment. The initial locations correspond to the epitope or the receptor of the antigen. Characteristics of the mines are stationary, unfriendly and competitive. Circling the mine is defined as diffusing the mine.

6.1.2 *SAGs* and its characteristics

Define the *B* cells to be the self-agents (*SAGs*). Deploy all the *SAGs* in a uniform distribution within the environment. The initial locations of the *SAGs* correspond to the receptors of the *B* cells. Characteristics of the *SAGs* are itinerant, independent, cooperative, altruistic and deliberative.

In mine detection application, it is assumed that the environment is accessible and the self-agents get updated information about the environment.

Assume that all the *SAGs* have the capability to sense the mine and communicate between the agents within the sensory and communication circles respectively. We have used Euclidean distance measure for both the cases. Every *SAG* (robot) recognizes the mine and identifies the location of the mine within this sensory circle. On identification of the *NAG* (mine) every *SAG* communicates to the other *SAGs* in a Jerne's network. For this problem, we have assumed Jerne's network as a broadcast network. It is also assumed that the communication between the *SAGs* is larger than the capacity of every *SAG* to sense the *NAG*.

6.1.3 Pseudo Code For The Mine Detection Problem

The pseudo code for the mine detection problem is as follows.

1. Initialize the *SAGs* and *NAGs* in a uniform distribution.
2. diff_use = 0; (Initially there is no diffusion)

2.1 **While (**diff_use ≠ number of_mines, N_i**)**

2.2 **For** each *SAG* S_j, do the following

If (there is a mine within the sensory circle)

a) Identify the location of the mine

b) Inform the locations of the mines to the other self-agents within the communication circle. This corresponds to the communication through the immune network.

c) *SAG* generates new actions that are eight different new locations to move

d) Find out the distance (affinity function) between these locations and mine locations. The Affinity is calculated by the Euclidean distance between the generated locations and the robot location.

e) Choose the distance that is lesser than an affinity threshold and move to that location.

f) **If** (this location is the mine location)

If (there are 4 *SAGs* around the mine)

Diffuse the mines, update the number of mines diffused, (diff_use = diff_use + 1);

If (diff_use == number of mines),

Break; End If; End While

STOP

Else wait until there are four *SAGs* around the mine; **End If**

Else do step 2.2. c. End If

Else If (there are any self-agents within the Communication circle)

- **If** (non-self information is available) repeat from step 2.2. **End If**

Else Make random movements from the current location, since there is no *NAG* information from other self-agents and no mine detected within the sensory circle

End If; End For; End While; STOP

Memory is not used in this problem since there is no usefulness in remembering the location of the mine once it is detected and diffused.

6.1.4 Simulation Results

We assume that a priori knowledge of the minefield intensity is known in the given environment. In the simulation, this means that the number of mines in the given environment is known. Therefore known number of mines is deployed in a uniformly distributed manner in the given area. This creates the minefield. We also deploy a known number of mobile robots in a uniformly distributed manner in the environment. The simulation differentiates the mobile robot and the mine by using a '+' for a mine and a 'o' for robots for representation while the code identifies a mine by a '0' and the robot by a '1'. The information vector for the mine and the robots contain the initially deployed location information along with the identifier. Table 1 shows an example of the mine and the robot information vector. The simulation also requires setting the sensory circle of the robot and the

communication circle. We have assumed that the communication circle is greater than the sensory circle.

Table 1: An Example of Information Vector of Mines and Robots

	X coordinate	Y coordinate	Identifier
Mine	4	5	0
	3	7	0
Robot	2	3	1

The simulation is verified for the following variations.

- By increasing the sensory range from 3 to 9 units of distance measure.
- The communication circle was varied between 5 and 11 units of distance measure.
- Changing the environments area to 10 x10 and 32 x 32 rectangular grids.

Here, the environment is accessible where each *SAG* has the information about the mines and the other *SAGs* in the sensory and communication neighborhood. That is, on identification of the mine, *SAGs* within the communication circle exchange about the number of mines detected and their respective locations through the agent broadcast network. A sample environment vector is shown in Table 2. It can be seen from Table 2 that the robot 1 has the information about mine 1 that is accessible to robot 2 if it is within the communication circle because robot 2 checks for the information available with robot 1 since it has not identified any mines. However, the environment becomes inaccessible on the assumption that the environment is not updated or when the communication circle is zero (c_cir = 0). It is useful to make the environment accessible in practice because, the mobile robots for mine detection can be provided with the capability to communicate.

Table 2: An Example of the Environment Vector

Index	Coordinates (Initial) X	Y	Identifier	No of mines detected	Detected Mine locations
Mines 1	3	7	0	0	--
2	4	5	0	0	--
Robots 1	2	4	1	1	4,5
2	5	2	1	0	--

The experiment is repeated for different populations of mines and robots. The typical range for the mines deployed are varied between 10 and 70 and accordingly and the robots are varied between 40 and 100. Figures 4 and 5 show the simulation with mines and robots with their initial locations and the four agents surrounding the mine. The following results prove that *AISIMAM* is able to solve the mine detection problem successfully.

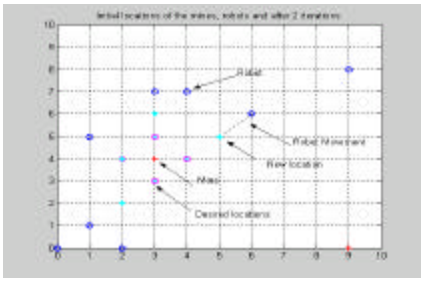


Figure 4: The locations of mines and robots after 2 iterations

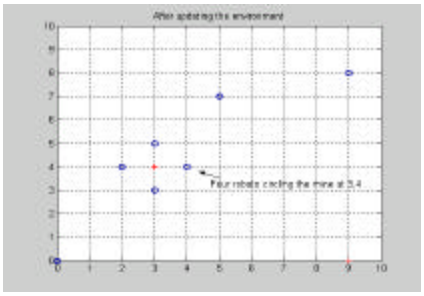


Figure 5: Four robots have circled one mine after three iterations

6.1.5 Observations

The following cases are studied and results are shown below.

a) For an increase in the population of mines and increase in population of robots the computational complexity in terms of rate of convergence (or the number of steps needed for the algorithm to converge) is studied. For an environment size of 32x32 and a constant sensory and communication circles, the individual rates of convergence are shown in Figures 6 and 8 and the average convergence rate can be seen in Figures 7 and 9. In Figures 6 to 9, x-axis is the number of mines, y-axis is the number of agents and z-axis is the number of iterations.

b) For an increase in the sensory region and communication region the computational time in terms of rate of convergence is studied. Increasing the sensory and communication circles reduce the required the number of steps for the algorithm to converge. This is due to the fact that robots senses more area and can communicate with more robots and check if others have mine information if they cannot find any.

The experiment is repeated for the same number of mines and number of robots with a step increase in the sensory and communication circles in the following combinational pairs (3,5), (5,7), (7,9) and (9,11). The number of iterations for a chosen value of robots and mines can be seen in Figures 6 and 8. The Figures 7and 9 shows the average number of iterations for sensory and

communication circles to be (5,7) and (7,9). However it was observed that increasing the sensory and communication circle reduces the average number of iterations for the algorithm to converge.

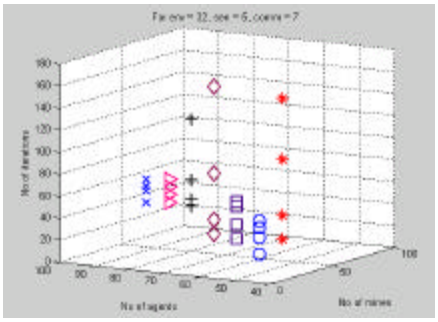


Figure 6: The rate of convergence for variation in mines and agents for 32x32, sen_c = 5, c_cir = 7

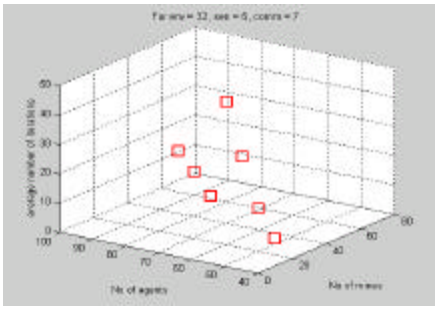


Figure 7: The average rate of convergence for variation in mines and agents for 32 x 32, sen_c =5, c_cir = 7

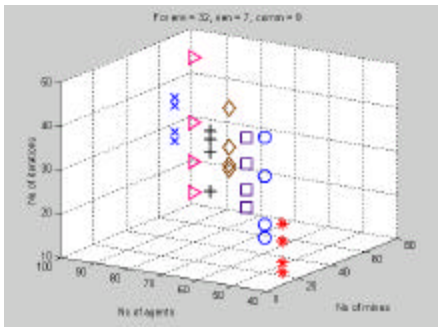


Figure 8: The rate of convergence for variation in mines and agents for 32 x 32, sen_c = 7, c_cir = 9

7 New Aspect of the Work

Literature survey shows that there are several applications on Artificial Immune Systems and Multi Agent Systems independently. Some of the recent work also addresses some of the properties of AIS to agent systems to solve a particular task (K. Mori, M. Tsukiyama and M. Fukuda 1997, D. Dasgupta 1998). *AISIMAM* is a generic model that provides to define the *SAGS* and *NAGS* in terms of functions to be determined by the applications. Individual goals and a global goal for the agents can also be defined by the functions. The model is flexible and unique because the parameters of the model can be changed by the formulated functions depending on the application.

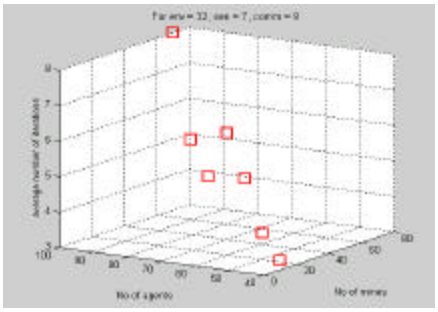


Figure 9: The average rate of convergence for variation in mines and agents for 32x32, sen_c =7, c_cir = 9

8 Future Work

A mathematical representation of the immune network is expected to be added in the future. Further conclusions can be arrived from the following additions. In the mine detection application,

a) We have assumed that the robots themselves do not get destroyed in the detection and diffusion process. But in practice, a robot can fall on the mine during deployment. So in future, the algorithm can be modified to analyze the case of robot falling on the robot while deployment and call that *failure rate analysis*.
b) Another assumption is that the *NAGs* or the mines in this application are static. This is true because in practice all the mines are static. In future applications, the *NAGs* could also be dynamic and hence the experiment can be repeated for the agent behavior.
c) Also, in the mine detection application, the memory is not used. This is because, there is usefulness in remembering either the location information of the mine or the type of mine itself. In future, we can redefine the application more specific by employing different functions for different kinds of mine. In this process, memory will be helpful in remembering the information about the type of mine that could be useful rather than the location information.

9 Conclusion

This research draws a generic model named *AISIMAM* based on artificial immune system applicable to intelligent multi agents. An application for the model is simulated. The mine detection and diffusion problem is experimented and the results show that *AISIMAM* is successful. The motivation for this application is that in future the mine detection can be performed efficiently by deploying mobile robots that have enough intelligence, communication and coordination to detect and diffuse the mines. To verify the generality of the model, more

applications will be simulated and verified in the future. This research is conducted with the support of Gleason R&D Funds in Multi-agent Bio-Robotics Lab (MABL) at Rochester Institute of Technology.

10 References

O. Abul, F. Polat and R. Alhajj (2000). "Multiagent Reinforcement Learning using Function Approximation", *IEEE Transaction on Systems, Man and Cybernetics*, Part C: Applications and Reviews, Vol 30, No 4, November.

A. H. Bond and L. Gasser (Eds.), (1988). *Readings in Distributed Artificial Intelligence*, Morgan Kaufman Publishers Inc, San Mateo, California, USA.

F. M Burnet (1978). "*Clonal Selection and after*", In Theoretical Immunology, (Eds.), G. I. B. S. Perelson and G.H. Pimbley Jr., Marcel Dekker Inc, pp 63 – 85.

L. N Castro, and F.J Von Zuben (1999). "Artificial Immune systems: Part I, Basic Theory and Applications", Technical Report – RT DCA 01/99, FEEC/UNICAMP, Brazil, 95 p.

L. N Castro and Von Zuben (2001). "aiNet: An Artificial Immune Network for Data Analysis", http://www.dca.fee.unicamp.br/~lnunes/publicat.html.

K. H Cho and J. Tae Lim (2001). "Multiagent Supervisory Control for Anti fault Propagation in Serial Production Systems", *IEEE Transactions on Industrial Electronics*, Vol 48, No 2, April.

D. Chowdary (1999). " Immune network : An example of Complex Adaptive Systems", Part II, pp 89–105, *In Artificial Immune Systems and their Applications*, Springer-Verlag, Heidelberg, Germany, 1999.

D. Dasgupta and N Attoh-Okine (1997). " Immunity Based Systems: A survey", *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp 363 – 374, Orlando, Florida.

D. Dasgupta (1998). "An Artificial Immune System as a Multi Agent Decision Support System", *Proceedings of the SMC98*, IEEE *international Conference on Systems, Man, and Cybernetics*, Vol 4, pp 3816-3820, Sandiego, Califronia.

D. Dasgupta (1999). *Artificial Immune Systems and Their Applications*, Springer-Verlag, Germany.

D. J. Smith, S. Forrest and A. S. Perelson (1996). "Immunological Memory is Associative", Part II, pp 105 –112, *Artificial Immune Systems and Their Applications*, Springer-Verlag, Germany.

K. D Elgert (1996). *Immunology - Understanding the Immune System*, John Wiley & Sons, Inc, NY, USA.

P. Hajela and J. S Yoo (1999). "Immune Network Modeling in Design Optimization", In *New Ideas in Optimization*, (Eds.) D. Corne, M. Dorigo & F. Glover, pp. 203-215, McGraw Hill, London.

A. Ishiguro, Y. Watanabe, and T. Kondo (1997). "A Robot with a Decentralized Consensus-Making Mechanism Based on the Immune System", In *Proc. ISADS'97*, pp. 231-237.

G. W. Hoffmann (1986). "A Neural Network Model Based on the Analogy with the Immune System", *Journal of Theoretical. Biology*, 122, pp. 33-67, 1986.

S. A. Hofmeyer and S. Forrest (2000), "Architecture for an Artificial Immune System", http://www.cs.unm.edu/~steveah.

M. N. Huhns and M. P. Singh (1998). *Readings in Agents*, Morgan Kaufman Publishers Inc., San Francisco, California, USA.

M. N. Huhns and L. M Stephens (1999). "Multiagent Systems and Societies of Agents," *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligenc*e, Gerhard Weiss (Eds.), MIT Press, Cambridge, Massachusetts, USA.

J. E. Hunt, and A. Fellows (1996). "Introducing an Immune Response into a CBR system for Data Mining", In *BCS ESG'96 Conference* and published as Research and Development in Expert Systems XIII, 1996.

N. K. Jerne (1984). "Idiotypic networks and Other Preconceived Ideas", *Immunological review*, Vol.79, pp.5-24.

A. M. Mohammed (2000). "Benevolent agents, PhD Thesis", Department of Electrical and Computer Engineering, University of South Carolina, USA.

K. Mori, M. Tsukiyama and M. Fukuda (1997). "Artificial Immunity Based Management System for a Semiconductor Production Line", *Proceedings of the SMC2001, IEEE international Conference on Systems, Man, and Cybernetics*, Vol 1, pp. 851 – 855, Atlanta, Georgia, USA.

H. S. Nwana and D.T. Ndumu (1997). "An Introduction to Agent Technology", *Software Agents and Soft Computing*, H. S. Nwana and N. Azarmi (Eds.), Springer-Verlag, Berlin, Germany.

S. Sathyanath and F. Sahin (2001). "Artificial immune Systems Approach to a Real Time Color Image Classification Problem", *Proceedings of the SMC2001*, IEEE *international Conference on Systems, Man, and Cybernetics*, Vol 4, pp. 2285 – 2290, Arizona, USA.

J. Timmis, J, M. Neal and J. Hunt (1999). " An Artificial Immune System for Data Analysis", *Proceedings of the International Workshop on Intelligent Processing in Cells and Tissues (IPCAT)*, Indianapolis, U.S.A,

S. Russell and P.Norvig (1995). *Artificial Intelligence: A Modern Approac*h, Prentice Hall, New Jersey, USA.

M. Wooldridge (1999). *Multiagent systems: a Modern Approach to Distributed Artificial Intelligence*, Gerhard Weiss (Eds.), The MIT press, Massachusetts, USA.

Immunocomputing for Bioarrays

Alexander O. Tarakanov
St. Petersburg Institute for Informatics and Automation,
Russian Academy of Sciences,
14-line 39, St. Petersburg, 199178,
Russia

Larisa B. Goncharova
Institute Pasteur of St. Petersburg,
Mira 14, St. Petersburg, 197101,
Russia

Tatyana V. Gupalova
Institute of Experimental Medicine,
Russian Academy of Medical Sciences,
St. Petersburg , Russia

Sergei V. Kvachev and **Alexander V. Sukhorukov**
Transas Co.Ltd,
St. Petersburg, Russia

Abstract

This paper presents results of application of our immunocomputing method to immune diagnostic arrays. The method detects bound complexes of immunoglobulin G (IgG) with protein G (pG), and recognizes the concentration of IgG as the result of IgG-pG interactions at each location of a bioarray. This model system has been developed as a prototype of a protein biochip for immunoassay-based diagnostics, where bioarray is a macro-variant of the biochip microarray, while the software is a core of the biochip reader and controller.

1 INTRODUCTION

By using optical densitometry or CCD (charge-coupled device) imaging system, it is possible to compare the differential protein levels among multiple samples. However, large-scale high-throughput methods of molecular immunology require rather complicated information processing to collect, analyze and interpret data. The problem becomes especially important for modern biochip technologies.

Any device which incorporates bioreceptors, such as antibodies, enzymes, cellular components of living systems etc., is referred to as a biosensor. Moreover, any biosensor that involves the use of a microchip system for detection is considered a biochip (Stokes et al., 2001).

Biochips (biological microchips or microarray technique) as the development and application of the arrays of immobilized biological compounds have become a significant trend in modern biology, biotechnology and medicine. The main advantage of biochips over conventional analytical devices is the possibility of massive parallel analysis. Biochips – really new and highly innovative products – are biological equivalents for computer microchips and they appeared as a result of application of the ideas of miniaturization, integration and parallel processing of information from microelectronics, where they were born, to biological processes. The chip principle has now become the dominating theme for a number of new proteomics technologies. Based on this principle, two main systems are currently used for analysis of multiple protein expression: two-dimensional polyacrylamide gel electrophoresis coupled with mass spectrometry and surface-enhanced laser desorption and ionization (Huang, 2002). However, the requirement of sophisticated devices often limits accessibility of these systems. At the same time, numerous proteins can be detected simultaneously and specifically using more simple immunoassay-based protein array systems. This approach (immunoassay) can be used to detect multiple proteins, including antibodies and antigens, toxins etc. Immunoassay-based microarray technologies can be particularly useful in accurately measuring the difference in individual protein levels between several samples, which is sometimes very important in disease monitoring.

On the other hand, we have developed a pattern recognition method based on our immunocomputing approach (Tarakanov, Skormin and Sokolova, 2002). Inspired by principles of antibody-antigen recognition in the natural immune system, our method solve problems of distinguishing background ("self") from patterns ("non-self") and processing the patterns. We have developed a rigorous mathematical basis and a software implementation of the method. The method has been applied to compute ecological atlases, predict danger of the plague infection, detect intrusions in computer networks, etc. (Kuznetsov et al., 1999; Tarakanov et al., 2000; Tarakanov and Skormin, 2002). In the present paper we apply our method to immune diagnostic arrays. We detect bound complexes of IgG with pG, and recognize the concentration of IgG as the result of IgG-pG interactions at each location of a bioarray. This model system has been developed as a prototype of a protein biochip for immunoassay-based diagnostics, where bioarray is a macro-variant of the biochip microarray, while the software is a core of the biochip reader and controller.

2 BIOARRAYS

2.1 GENERAL APPROACH

In this work the sampling platform is a nitrocellulose membrane exposed to different concentrations of IgG and subsequently analyzed using a direct immunoassay involving pG labeled by horseradish peroxidase (pG-HRP) or carbon particles (pG-CP).

To quantify the exact amount of proteins, multiple standard curves can be generated and according to them the exact amount of individual proteins can be detected. We used the twofold dilution standard curve as an example (model) for biological experiments.

The ability of the surface proteins of Streptococci, pG among them, to interact with human and animal plasma and serum proteins is well known and has gained a prominent place in immunochemistry. Immunoglobulins interact with the surface receptor of the microorganism by means of their Fc-fragment, that allows to use widely the bacterial pG in all clinical assays when it is necessary to detect serum antibodies. We use pG-IgG interacting system to obtain the protein arrays in the direct immunoassay.

2.2 MATERIALS AND METHODS

Nitrocellulose membranes for protein arrays ("dot blots"), pore size 0,45 μm, have been purchased from Millipore Corp., USA, Filter type HA, Cat.No HAHY 304 FO.

Human IgG (Fractions II, III) has been purchased from Sigma, USA.

Recombinant pG of group G Streprococci, strain G148, has been cloned at the Institute of Experimental Medicine, Russian Academy of Medical Sciences, St-Petersburg, Russia (Gupalova and Totolian, 1996). The corresponding HRP-conjugated pG (pG-HRP) has been prepared by periodate method (Frimel, 1987).

The detection reagent of recombinant pG covalently conjugated to particles of colloidal carbon (pG-CP) ranging from 150 to 200 nm in size, has been kindly gifted by Dr. M.B. Raev, Institute of Ecology and Genetics of Microorganisms, Russian Academy of Sciences, Perm, Russia (Plaksin et al., 1996).

2.3 PREPARATION OF BIOARRAYS

2.3.1 Spot

We use nitrocellulose membranes to spot the twofold dilutions of IgG.

To spot capture proteins IgG onto membranes, we place the nitrocellulose strips on the top of white light box. We load manually 5 μl of solution of IgG (the initial concentration of IgG is 250 μg/ml) onto a single spot by a 10-μl pipettors.

2.3.2 Immunoassay

We load human IgG onto membranes as described above.

We block membranes with "blotto" solution (1% non-fat milk, pH 7,2) for 1 hour at room temperature. "Blotto" is used as washing and dilution solution at all steps, if necessary.

In case of usage pG-CP as the detection reagent, the 5-10 minutes detection procedure with conjugate follows after the 20-30 minutes incubation period.

The results are visualized as black dots directly on a white solid phase surface of nitrocellulose membrane. There are no additional steps.

In case of usage of pG-HRP conjugate, we incubate membranes for 1 hour at room temperature and use α-naphthol as the substrate for HRP. We dilute 6 mg of α-naphthol in 2 ml of methanol; we add 8 ml of 0.01 M Tris pH 7.5/0.5 M NaCl and 6 μl of 30 % hydrogen peroxide to the final solution.

2.3.3 Solid-phase immunoassay

In this study we have implemented a direct immunoassay method using pG-labeled probes. We use nitrocellulose membranes (strips) for the assays. We spot different concentrations of human IgG onto membranes as the model of "calibration curve": {250, 125, 62.5, 31.2, 15.6, 7.8, 3.9, 1.95, 0.99, 0.5} μg/ml.

We dry the membranes at room temperature for 10 minutes and then place them in blocking solution of "blotto" to block any unoccupied binding sites on the membrane surface for 1 hour at room temperature. This step is followed by removal of the blocking solution. We prepare dilutions of pG-HRP or pG-CP (usual dilutions are 1:2000 and 1:25) in the same buffer as for the blocking step. After incubation of the membranes in the conjugate solution for 0,5-1,0 h at room temperature, we remove the solution and wash the membranes 5 times using 0,5 % Tween 20 in PBS.

When the pG-CP conjugate is used in the assay, there are no additional steps and the results of the assay are visualized as black dots directly on the nitrocellulose surface. The obtained results are stable and require no specific fixation or termination procedures.

When the pG-HRP conjugate is used in the assay, there is an additional step of the enzyme-substrate reaction with α-naphthol, as described above (see Section 2.3.2.).

2.3.4 Experimental results

We obtain bioarray as a nitrocellulose strip with the rows and columns of IgG. Figure 1 shows an example of the bioarray. Typical size of the bioarray is 15×70 mm approximately. We have prepared about 50 of such bioarrays.

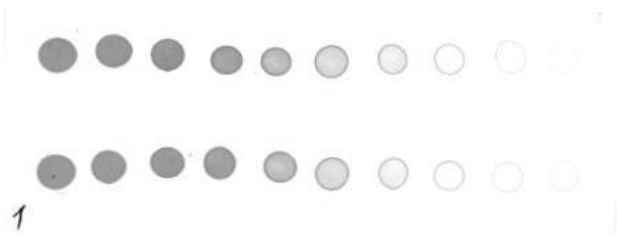


Figure 1: Bioarray of pG-IgG

3 SOFTWARE

3.1 COMPUTER INPUT

We scan bioarrays by a custom scanner HP ScanJet 5300C to make them available for processing on any usual Personal Computer (PC). The resolution is 150 dpi in color format. Conventional files in the bit map pixel format (bmp) are obtained. We assign names to these files according to the following format:

YearMonthDateGroupNumber,

where Group = {01, 02}, Group = 01 corresponds to pG-HRP, and Group = 02 corresponds to pG-CP conjugates. For example, the name of the bioarray's file in Figure 1 is as follows:

0205140101.bmp,

which means May 14, 2002, pG-HRP conjugate, bioarray #01.

A database prototype has been developed as a file in the Exel format (xls). The records of this database correspond to the bioarrays, where file name is a key to the record. Another data correspond to conditions of bio-membranes experiments, and results of image recognition.

3.2 IMMUNOCHIP EMULATOR

Software for image recognition by immunocomputing has been developed as a version of software emulator of an immunochip (Tarakanov and Dasgupta, 2002). The emulator is being developed under the following standard software: MS Windows'2000 operating system, MS Visual C++ 6.0 Developer Studio, and OpenGL graphic tools. Figure 2 shows a screenshot of the emulator.

The emulator works as follows.

User opens file with an image of bioarray (e.g., as in Figure 1). The image appears in the left-hand screen of the emulator.

User clicks the "Run Processing" button on the emulator toolbar. The emulator recognizes the locations (spots) of the immune reactions by distinguishing them from the background of the image. Such spots are outlined by the emulator (see squares in left-hand screen of Figure 2).

Then each spot is processed by the emulator to recognize the concentration of IgG. The corresponding concentrations are shown as the up-centered table in Figure 2.

Right-hand screen graphics of the emulator in Figure 2 represent inner parameters of the method. They are used by developers to control image recognition processes, as described below.

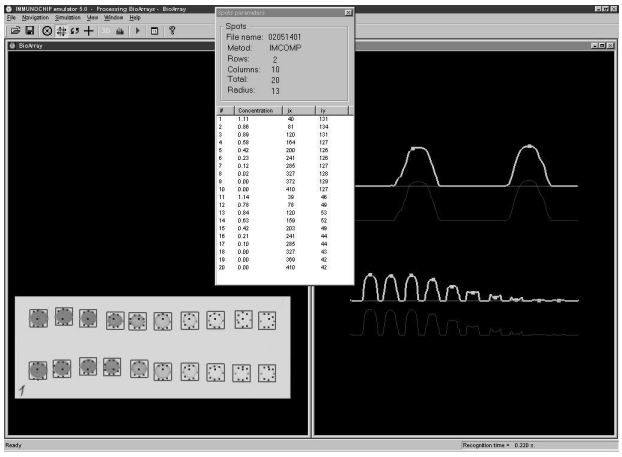


Figure 2: Immunocomputing Software

3.3 MATHEMATICAL BASIS

Bioarray image is represented initially as a matrix M_A of dimension $n_L \times n_R$. According to the (bmp) format, we form this matrix as a matrix of real values:

$$m_{ij} = (r_{ij}+g_{ij}+b_{ij})/3,$$

$$M_A = \{m_{ij}\},\ 0 \le m_{ij} \le 255,\ i = 1,..., n_L,\ j = 1,..., n_R.$$

where r, g and b correspond to the red, green and blue values of the pixel.

According to the "key and lock" principle of antibody-antigen recognizing in the immune system, we consider the matrix M_A as a collection of "antigens-locks". To compute "antibodies-keys", we form an inverse matrix M of the same dimension:

$$M = 255 - M_A.$$

We represent this matrix in the following form:

$$M \cong sLR^T,\ L^TL = 1,\ R^TR = 1, \qquad (1)$$

where s is first singular value; L and R are first left and right singular vectors of the matrix M.

It is known (see e.g., Tarakanov and Skormin, 2002) that representation (1) corresponds to the first term of the so-called Singular Value Decomposition (SVD). Such SVD exists for any rectangular matrix over the field of real values and can be computed by the following iterative scheme:

$$L^{(1)} = [\ 1, ..., 1\ ]^T,\ L^{(1)} = L^{(1)} / | L^{(1)} |,$$
$$[R^{(k)}]^T = [L^{(k-1)}]^T M,\ R^{(k)} = R^{(k)} / | R^{(k)} |,$$
$$L^{(k)} = MR^{(k)},\ L^{(k)} = L^{(k)} / | L^{(k)} |,$$
$$s^{(k)} = [L^{(k)}]^T MR^{(k)},$$
$$k = 2,..., \text{until } | s^{(k)} - s^{(k-1)} | < \varepsilon .$$

According to (Horn and Johnson, 1986), such a scheme converges to the maximal singular value and singular vectors in general case of the matrix M.

First singular value s and corresponding singular vectors L, R of the matrix M possess the following property:

$$s = L^T MR,\ s \geq P^T MQ,\ \ \forall P,Q: P^T P=1,\ Q^T Q=1 .$$

In other words, representation (1) of any real matrix M is mathematically optimal (in the sense of the minimal least square error), if, and only if, s is the first singular value, and L, R are the corresponding singular vectors of the matrix.

3.4 RECOGNITION ALGORITHMS

According to (Tarakanov and Skormin, 2002), unit vectors L, R can be considered as a mathematical model of "antibodies-probes", while $w=-s$ is their binding energy within so-called Formal Immune Network (FIN). By such a way, using our immunocomputing approach, we reduce two-dimensional input "antigen" M_A to two one-dimensional "antibodies" L, R. These "antibodies" represent a kind of "internal image" of the "antigen", generated by FIN. Figure 2 shows the profiles of such "antibodies" in the right-hand screen.

Using decomposition (1), we reduce the problem of detecting spots in the input image M_A (left-hand screen in Figure 2) to more simple task of finding local minima of one-dimensional functions L, R (right-hand screen in Figure 2). The emulator determines these minima as "paratopes" (antigen binding parts of antibody), according to an analogy of "antigen processing" by the natural immune system.

For example, designate vector L in the k step of processing as $L(k)$, $k=1,...,k_m$, where k_m is a number of local minimum. Hence, initially $L(1)=L$.

Determine global minimum of $L(1)$ and sequentially cut off the corresponding local "paratope" $L_P(1)$ by small pieces, until a "self" level of the background be obtained. Figure 3 illustrates such processing of the right-hand "paratope".

By such a way, the emulator computes roughly a position of the center and the bounds of the second row of the spots of the bioarray (as shown in left-hand screen of Figure 2).

Then the emulator cuts off the found "paratope" and repeat processing on another step, until all "paratopes" be found:

$$L(k) = L(k-1) - L_P(k-1),\ k = 2,...,k_m .$$

Figure 3: Processing of the "Paratope"

Such "paratopes" of the left singular vector L correspond to the rows of the bioarray. For example, if vector L has two "paratopes" (as in Figure 3 and in the upper graphics of right-hand screen in Figure 2), then the bioarray has two rows of spots (as in left-hand screen of Figure 2).

Analogously, the emulator finds all "paratopes" of the right singular vector R. They correspond to the columns of spots of the bioarray. For example, if vector R has ten "paratopes" (as in the lower graphics of right-hand screen in Figure 2), then the bioarray has ten columns of spots (as in left-hand screen of Figure 2).

As a result, the emulator detects roughly positions of all spots of the bioarray as the squares (see left-hand screen of Figure 2).

After that, the emulator defines more exactly positions and sizes of the spots within the squares. According to "key and lock" principle of "antibody-antigen" binding, the emulator adjusts the shape of the "paratope" by maximal overfall between the brightness of the neighboring pixels of the "antigen" within every square. This step of processing uses two directions: rows i and columns j.

Let i_C , j_C are coordinates of the center, and r is half-side of the square in pixels. The adjusted bounds of the spot i_a , i_b , j_a , j_b , are determined by the following way:

$$i_a: max\ \{m_{i,j} - m_{i+1,j}\},\ \ i_b: max\ \{m_{i+1,j} - m_{i,j}\},$$
$$j = j_C ,\ i = i_C - r, ..., i_C + r,$$
$$j_a: max\ \{m_{i,j} - m_{i,j+1}\},\ \ j_b: max\ \{m_{i,j+1} - m_{i,j}\},$$
$$i = i_C ,\ j = j_C - r, ..., j_C + r.$$

The process repeats with the new values:

$$i_C = (i_a + i_b)/2 ,\ \ j_C = (j_a + j_b)/2 ,$$
$$r = max\{(i_b - i_a), (j_b - j_a)\}/2 ,$$

until the difference between previous r and the new one becomes no more than two pixels. Figure 2 shows results of such adjusting in left-hand screen, where emulator has marked each detected spot by bold points.

After the spots are detected and adjusted, the emulator recognizes the concentration of IgG in each spot using SVD representation (1) of the spot image.

Let M_S be matrix of the spot image:

$$M_S \subset M,\ \ M_S = \{m_{i,j}\},$$
$$i = i_C - r, ..., i_C + r,\ \ j = j_C - r, ..., j_C + r .$$

According to (1), the emulator computes first singular value s of this matrix, and consider the concentration as follows:

$$C(IgG) = c_e\, s_m\,,$$

$$c_e \cong 1000,\ \ s_m = s / (4r^2)\,,$$

where s_m is mean singular value per pixel of the spot, and c_e is an experimental coefficient of converting brightness to concentration.

3.5 RECOGNITION RESULTS

Numerical experiment has been staged using the test set of 19 bioarrays: 0205140101 – 0205140119.

Any bioarray has 2 rows and 10 column of spots (see left-hand screen in Figure 2). Hence, full number of spots to be recognized is equal to:

$$19\times(2\times10) = 380\,.$$

Table 1 shows results of recognition of the spots.

Table 1: Number of Undetected Spots

Bioarray	**Undetected Spots**		**Recognition time (sec)**	
02051401#	**IMCOMP**	**MASK**	**IMCOMP**	**MASK**
01	0	0	0.36	3.32
02	0	2	0.41	3.12
03	0	1	0.20	2.66
04	0	3	0.32	2.86
05	0	2	0.22	2.28
06	4	0	0.23	2.19
07	0	0	0.26	2.22
08	6	1	0.27	2.62
09	0	1	0.26	3.25
10	0	1	0.30	2.22
11	0	2	0.28	2.92
12	0	0	0.24	2.65
13	0	2	0.42	2.60
14	0	2	0.22	2.64
15	0	1	0.34	2.23
16	0	0	0.28	2.19
17	0	0	0.23	2.03
18	0	0	0.26	3.01
19	4	3	0.29	2.21
Spots total	Total undetected		Mean time per spot	
380	14	21	0.014	0.130

We have also compared our immunocomputing method (IMCOMP) with another method of recognition by "mask" (MASK). Apparently, the MASK is the most traditional, direct and simple method of image recognition. Usually "mask" represents a square, which pixels form a sample of the object to be recognized within the image. The image is scanned by the "mask" to find any location, where the correspondence between the "mask" and the part of the image is no less than some threshold. The correspondence is computed by comparing all pixels of the "mask" and the part of the image, covered by the "mask". In our case the "mask" represents a sample of the spot.

According to Table 1, the recognition rate of the IMCOMP method is equal to

$$(1-14/380)\times100\% \cong 96.3\%\,,$$

while the recognition rate of the MASK method is slightly worse:

$$(1-21/380)\times100\% \cong 94.5\%.$$

Table 1 also shows that recognition by the IMCOMP method is almost 10 times faster than by the MASK method.

It worth noting, that all spots undetected by the IMCOMP method are almost invisible. Figure 4 shows a typical example of such undetected spots in the bioarray 0205140106.

As one can see from Figure 4, the last two spots in the right-edge column of the bioarray are almost invisible. As a more cogent argument, the profile of the right singular vector is given below the spots. It is obvious, that the last right-hand "paratopes" of the vector doesn' t differ from the background.

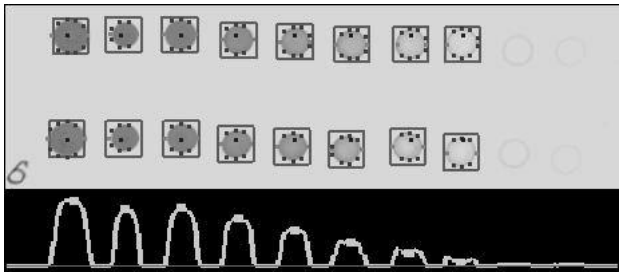


Figure 4: Example of Undetected Spots

Two other numerical experiments have been staged to recognize the concentration of IgG in the spots. Tables 2, 3 and 4 show results of the experiments, where the expected IgG (μg/spot) is approximately equal to 1:200 (ml/spot) of the diluted IgG (μg/ml). For example, 1.25 μg/spot corresponds to 250 μg/ml diluted, 0.63 μg/spot corresponds to 125 μg/ml diluted, etc.

Table 2: Two Testing Bioarrays

Bioarray	Size (pixels)	Spots' Radius (pixels)
0203070101	152×598	11
0205140101	167×450	12

Table 3: Spots of Bioarray 0203070101

Spot # (row – column)	Center i-j (pixels)	IgG Recognized (μg/spot)	IgG Expected (μg/spot)
1-1	25-22	1.24	1.25
1-2	24-75	0.78	1.25
1-3	24-135	0.55	1.25
1-4	25-190	0.52	0.63
1-5	24-246	0.41	0.31
1-6	23-298	0.40	0.16
1-7	24-353	0.13	0.08
1-8	21-407	0.06	0.04
1-9	23-459	0.02	0.02
1-10	22-515	0.00	0.01
2-1	132-20	1.28	1.25
2-2	132-78	0.82	1.25
2-3	133-132	0.58	1.25
2-4	132-190	0.50	0.63
2-5	133-246	0.47	0.31
2-6	131-297	0.22	0.16
2-7	130-351	0.08	0.08
2-8	131-407	0.03	0.04
2-9	138-461	0.05	0.02
2-10	131-517	0.01	0.01

It worth to admit, that recognition of the IgG concentration in spots needs to be seriously improved. However, it has to be noted that the expected IgG quantity in each spot may differ from the real IgG quantity in the spot due to the heterogeneous binding capacity of nitrocellulose membrane. Moreover, the real IgG quantity in the spot depends mainly from the accuracy and purity of biological experiments, rather than from the accuracy of our recognition method. The primary problem is to get an objective data on the concentration of IgG in the obtained bioarrays.

Table 4: Spots of Bioarray 0205140101

Spot # (row – column)	Center i-j (pixels)	IgG Recognized (μg/ml)	IgG Expected (μg/spot)
1-1	45-39	1.17	1.25
1-2	49-77	0.96	1.25
1-3	52-120	0.94	1.25
1-4	51-158	0.69	0.63
1-5	49-202	0.53	0.31
1-6	44-240	0.27	0.16
1-7	45-285	0.19	0.08
1-8	43-326	0.02	0.04
1-9	41-368	0.05	0.02
1-10	43-409	0.01	0.01
2-1	130-40	1.23	1.25
2-2	133-81	0.98	1.25
2-3	130-120	0.90	1.25
2-4	126-163	0.69	0.63
2-5	125-199	0.48	0.31
2-6	125-240	0.24	0.16
2-7	126-284	0.10	0.08
2-8	127-327	0.00	0.04
2-9	127-370	0.01	0.02
2-10	127-408	0.00	0.01

As an example, Table 5 shows optical density of the spots of the bioarray 0203070101. These data have been obtained by the UltraScan XL Laser Densitometer (LKB Bromma, Sweden). However, the values of the optical density have been read out visually, and the conversion of the optical density d to the concentration of IgG based also on an experimental dependence:

$$C(IgG) = (d - 2.1)/0.24 .$$

It worth noting, that the above results in Tables 1-4 have been obtained by the automatic mode of the emulator. In this mode the emulator recognizes spots in any bioarray without any prompting by user. This mode of the emulator corresponds to the unsupervised learning in terms of pattern recognition (Tarakanov and Skormin, 2002).

However, we have also developed a possibility for user to show spot to the emulator in so called training mode. This mode corresponds to the supervised learning (training) in terms of pattern recognition. Figure 5 shows an example of using this mode.

Table 5: Spots of Bioarray 0203070101 with Optical Density

Spot # (row – column)	Optical Density (d)	IgG Computed (μg/spot)	IgG Expected (μg/spot)
1-1	2.37	1.13	1.25
1-2	2.33	0.96	1.25
1-3	2.28	0.75	1.25
1-4	2.26	0.67	0.63
1-5	2.23	0.54	0.31
1-6	2.23	0.54	0.16
1-7	2.17	0.29	0.08
1-8	2.16	0.25	0.04
1-9	2.13	0.12	0.02
1-10	2.13	0.12	0.01
2-1	2.4	1.25	1.25
2-2	2.34	1.00	1.25
2-3	2.32	0.92	1.25
2-4	2.28	0.75	0.63
2-5	2.26	0.67	0.31
2-6	2.19	0.38	0.16
2-7	2.18	0.33	0.08
2-8	2.16	0.25	0.04
2-9	2.17	0.29	0.02
2-10	2.17	0.29	0.01

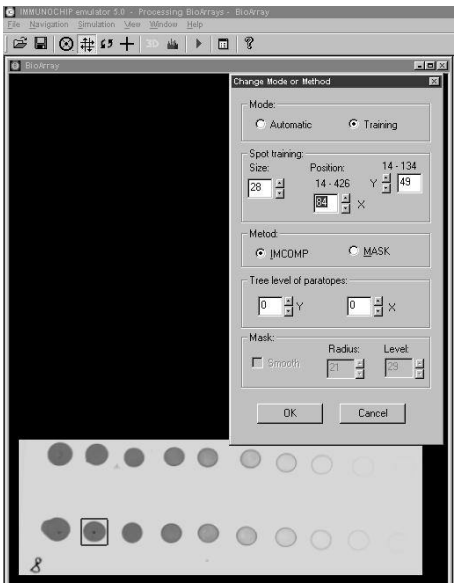


Figure 5: Example of Training the Emulator

In the training mode user points to a spot by means of the square in the left hand screen. By changing the size and the position of the square, user covers the spot. By such a way, the emulator learns about the size of the spots ("non-self") and the value of the background threshold ("self"). Numerical experiment has shown, that the emulator detects all undetected spots in Table 1 by using the training mode. Figure 6 shows an example of detecting all spots of bioarray 0205140106 by using the training mode. Note, that four spots of this bioarray haven' t been detected by using the automatic mode (see Figure 4).

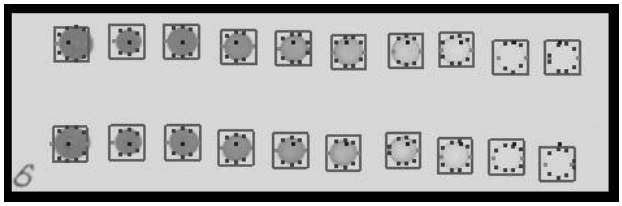


Figure 6: Detection of All Spots by Training Mode: Compare with Figure 4

It worth also noting, that the emulator detects spots, and only spots as "non-self" codes. For example, the emulator doesn' t detect any other codes like written numbers of biomembranes (see Figures 2, 4, 6). This property corresponds to so-called "self-tolerance" of the immune system.

4 DISCUSSION

4.1 ARTIFICIAL IMMUNE SYSTEM

The main goal of our work is to create an artificial immune system as a computer controlled fragment of the natural immune system. This paper presents results of the first step in this direction.

We use two "immune" methods in our work: direct immunoassay method to create bioarrays and immunocomputing method to recognize results of immune reactions.

As a fragment of the immune system in vitro we use bioarrays of immunoglobulin (IgG) molecules. These molecules are attached to the nitrocellulose membrane to form arrays of a kind of immune memory, like the silicon cells form arrays of computer memory. Using computer analogy, we can also consider, that concentrations of IgG in locations (spots) of bioarray correspond to the stored values in cells of computer memory.

To expose the concentration of IgG we use special protein (pG) as a testing molecular signal, like special electronic signals are used to read out contents of computer memory. The results of immune interactions between IgG and pG are visualized as black spots.

Therefore, main goal of our immunocomputing method is to recognize such spots on a surface of nitrocellulose membrane. In other words, we need this method to recognize the concentration of IgG in each location of bioarray.

Inspired by principles of antibody-antigen recognition in the natural immune system, our method solve problems of distinguishing background ("self" codes) of nitrocellulose membranes from spots ("non-self" codes) of bioarrays and processing the spots to recognize concentrations of immune molecules.

A core of our immunocomputing approach to pattern recognition is a mathematical model of binding, or recognizing between antibodies and antigens. According to the biological prototype, the central notion of this model is binding energy. We determine the binding energy by a bilinear form over the pair of corresponding vectors. This bilinear form is determined by a real rectangular matrix. Thus, we obtain a convenient quantitative measure of the extent of recognition between the antibodies and antigens, as well as a rigorous mathematical model of the recognition based on SVD of real valued matrices.

An advantage of our approach consists in reducing two-dimensional input image of biomembrane to two one-dimensional "antibodies". This feature allows to reduce recognition time drastically, as shown in Table 1.

Another advantage seems to be a possibility of an effective hardware implementation of the approach in so called immunochip (Tarakanov and Dasgupta, 2002). Such miniature silicon device could be very useful as a part of biochips for immune diagnostics.

4.2 BIOCHIP

We consider a biochip approach as a way to improve and to develop our bioarray technology in general (Tarakanov and Goncharova, 2002). To overcome the main deficiencies of the technology, we are developing the so-called Biochip Controlling System (BCS). The core of this system is the presented immunocomputing software. The development of BCS will allow to obtain the needed accuracy and purity of biological experiments. The function of BCS will be twofold: 1) as a liquid delivery system of the biochip and 2) as a scan reader of the biochip reactions.

Our BCS will use a combined method of the analysis of the biochip reactions based on photometric and imaging detection procedures. This method will be applied for the analysis of optical and spatial parameters of reaction locations on the biochip. Each location will include the carbon particles labeled reagent system (CP system). The photometric procedure will provide reflection measuring of the optical density of the CP system. This procedure will be basic for the analysis of the reactions by the BCS. The imaging procedure will detect the spatial parameters of the CP system. This procedure will be additional for the BCS. The combination of these two procedures should provide sure detection of the results of the biochip reactions.

The BCS will contain: a) precision motorized two coordinate stage; b) auto-manual or automated dispense module and c) precision optical electronic detection module.

The two coordinate stage and the dispense module will be controlled by a PC compatible computer. We suppose to use two types of dispense modules available in the market: auto-manual and automated. Accordingly, our BCS with the auto-manual dispense module could be used in medical laboratories, while that with the automated dispense module could be used for industry applications.

We intend to use a modification of the available electronic pipettor as the auto-manual dispense head. This pipettor is a multi-function instrument. It provides the reagent' s loading, diluting, mixing and dispensing with the high accuracy by means of the embedded microprocessor. The electronic pipettor will be moved manually to the pipette washing sub-module and the reagent' s loading sub-module. These sub-modules will be arranged separately.

The automated dispense module unites the dispense head, washing sub-module and loading sub-module in one system. All operations are performed by this dispense module automatically.

The functioning of the BCS includes two steps. First, the biochip will be placed on the horizontal surface of the two coordinate stage under the dispense head. The reagents will be dispensed on the biochip by means of its scanning relatively to the dispense head. Second, the biochip will be moved by means of the two coordinate stage to the detection module and scanned by the detection head.

The digital or analog signals from the detection module will be transmitted to the PC computer for processing through the standard serial port or a special PCI card.

An important application of such biochip can be early diagnostics of C-Reactive Protein as a reliable biochemical marker suitable for detection of tissue damage, necrosis and inflammation (Tarakanov and Goncharova, 2002).

4.3 BIOCOMPUTER

In perspective, two steps could transform the proposed biochip to a biomolecular computer: 1) an immunochip-based controller, including 2) controlling of biomolecules in the micro-wells of the biochip.

The first step is needed to replace a PC-compatible computer by an immunochip. At this step the function of the immunochip is twofold: 1a) control of the liquid delivery system of the biochip and 1b) control of the biochip reader.

The first step – the immunochip-based biochip controller – is necessary for control reactions in the micro-wells of the biochip. For example, the immunochip could

automate feeding the biochip with reagents and samples, removing intermediate products, changing probes in the process of training of the biochip, etc. Although such functions seem complicated, it is worth noting that they are currently under development for some microfluidic biochips (Huang, 2002). Also at this step the immunochip should provide a surveillance of the biochip surface, including image processing from the biochip reader and recognition of the results of the reactions in the micro-wells.

The second step presents a solution to the key problem of the biocomputer: providing control of biomolecules in the micro-wells of the biochip by a molecular-electronic impact computed and performed by the immunochip. Simply put, the biocomputer could secrete biomolecules with needed properties at appropriate locations on the biochip. If the problem is solved, the next step could be secretion of necessary biomolecules at appropriate times. For example, in this way the implanted biocomputer could control and correct natural immunity.

Therefore, the above two steps would allow us to obtain a full-value biocomputer, where natural biomolecules (proteins and DNAs) of the biochip collaborate with the silicon schemes of the immunochip.

It is worth noting that the choice of the C-Reactive Protein for one of the variants of the biochip is not accidental. The functions of this protein are close to those of cytokines – special proteins secreted by immune cells to control immune response. It is known that violation in synthesis and secretion of the cytokines could cause several violations of immunity. Therefore, the development of the biochip for detection of such proteins is also a step by the biocomputer for evaluation and control of the cytokine system in general. So, the development of the biocomputer to control cytokine complex in model biological micro-systems (in vitro) as a fragment of computer controlled immune system seems quite realistic and well-timed.

5 CONCLUSIONS

In our opinion, an important criterion of success of any biological inspired approach in mathematics or computer science is an effective application of the approach to the area where it had came from. Accordingly, we have made an attempt to apply our immunocomputing approach to immunoassay-based diagnostic arrays. We can consider our attempt as successful, because our method recognizes more than 96% spots as "non-self" codes. It works almost 10 times faster than direct recognition of spots by comparing them with a mask sample. A comparison of our recognition method with other approaches and its advantage had been also shown on the example of another immunological application (Tarakanov et al., 2000).

Acknowledgments

The development of mathematical methods and software of this work is partially supported by the European Commission under the EU project IST-2000-26016 "Immunocomputing", and the European Office of Aerospace R&D under the EOARD-ISTC project 2200P "Development of Mathematical Models of Immune Networks Intended for Information Security Assurance".

References

G. Frimel (1987). *Immunological Methods*. Moscow: Medicine (in Russian).

T.V. Gupalova and A.A. Totolian (1996). Recombinant IgG-binding protein of group G Streptococci. *The Russian Federation Patent # 2056859* (in Russian).

R. Horn and C. Johnson (1986). *Matrix Analysis*. Cambridge University Press.

R.-P. Huang (2002). Detection of multiple proteins in an antibody-based protein microarray system. *J. of Immunological Methods 255*, 1-13.

V.I. Kuznetsov, V.B. Milyaev and A.O. Tarakanov (1999). *Mathematical Basis of Complex Ecological Evaluation*. St.Petersburg University Press.

D.Yu. Plaksin, M.B. Raev and E.T. Gromakovskaja (1997). The method of stereospecific assay and the method of the conjugate for stereospecifis assay. *The Russian Federation Patent # 20899212* (in Russian).

D.L. Stokes, G.D. Griffin and T. Vo-Dinh (2001). Detection of E.coli using a microfluidic-based antibody biochip detection system. *Fresenius J. of Analitical Chemistry* 369: 295-301.

A. Tarakanov and D. Dasgupta (2002). An immunochip architecture and its emulation. *Proc. of the 2002 NASA/DoD Conf. on Evolvable Hardware EH-2002*. Washington, DC, USA.

A. Tarakanov and L. Goncharova (2002). Immunocomputing to biochip and biocomputer. *Proc. of the 6th World Multiconference on Systemics, Cybernetics and Informatics SCI-2002*. Orlando, Florida, USA.

A. Tarakanov, S. Sokolova, B. Abramov and A. Aikimbayev (2000). Immunocomputing of the natural plague foci. *Proc. of the 2000 Genetic and Evolutionary Computation Conference (GECCO-2000)*, 38-41, Las Vegas, USA.

A. Tarakanov and V. Skormin (2002). Pattern recognition by immunocomputing. *Proc. of the 2002 World Congress on Evolutionary Computation (CEC-2002)* **1**, 938-943. Honolulu, HI, USA.

A. Tarakanov, V. Skormin and S. Sokolova (2002). *Immunocomputing: Mathematical Basis and Applications*. Springer, New York (in press).

Evolving FPGA-based robot controllers using an evolutionary algorithm

Renato A. Krohling, Yuchao Zhou, and Andy M. Tyrrell

Bio-Inspired Eng. Lab.

Department of Electronics

University of York

Heslington,YO10 5DD

York, UK

E-mail: rk8@ohm.york.ac.uk

Abstract

In this paper, a novel evolutionary algorithm for intrinsic hardware evolution of Field Programmable Gate Array (FPGA) controllers is presented. The main feature of the evolutionary algorithm consists of a mutation operator, in which the mutation rate is defined according to the fitness. Experimental results on a Kephera robot show that the algorithm proposed can successfully navigate the robot to avoid collision in an unknown/changing environment.

1 INTRODUCTION

Autonomous robot navigation is a very challenging problem. Conventional approaches based on off-line learned control policies generally do not work appropriately when implemented in real time environments. The development of a new research field named Evolvable Hardware (EHW), i.e., application of evolutionary algorithms to automatic design/ reconfiguration of electronics circuits (Zebulum et al., 2001) presents a great potential to tackle the problem of adaptation in unknown/changing environments. It is proposed that autonomous robotics could benefit from the development of EHW.

Two methodologies have been established for the design of EHW: Extrinsic and intrinsic (Thompson, 1996), (Miller and Thompson, 1998), (Layzell, 1998), (Haddow and Tufte, 2000), (Hollingworth et al., 2000). In the former case, both the evolutionary process[1] as well as the fitness evaluation of each individual (the circuit) are simulated in software. The entire design is carried out off-line and after the evolutionary process has completed, the hardware is implemented in real time. In the latter case, the evolutionary process is simulated in software but each individual is executed in hardware in real time (on-line evolution) (Haddow and Tufte, 2000), (Hollingworth et al., 2000). This is possible due the development of electronic devices such as Field Programmable Gate Array (FPGA), which are reconfigurable devices with no pre-determined function (Shirasuchi, 1996). Each individual is represented as a Bitstring that is downloaded to the chip as configuration data. This data includes a definition of each cell's functionality as well as the topology of the system.

For autonomous robot navigation intrinsic evolution presents a promising approach. A standard Genetic Algorithm (GA) has been applied to EHW because of its binary representation, which matches perfectly with the configuration bits used in FPGA. Some work has been produced to evolve on-line FPGA-based robot controllers using genetic algorithms (Thompson, 1995), (Keymeulen at al., 1997), (Haddow and Tufte, 1999), (Tan et al., 2002). For on-line evolution the fitness is evaluated on target hardware. Therefore changes in environment are reflected immediately in the fitness evaluation. Unfortunately on-line evolution is time consuming, especially for robot navigation in an unknown environment. So it is necessary to impose restriction on the population size.

In this paper, the problem of real time adaptation of autonomous robot navigation in changing environment is formulated as a time-dependent optimization problem: Find an appropriate *function F* (the controller) which maps the inputs from sensors to the outputs (control signal to the motors). To solve this problem, a novel evolutionary algorithm to evolve a reconfigurable FPGA-based controller is proposed.

The paper is organized as follows: in section 2 a description of the algorithm is given; section 3 describes the Kephera robot, the hardware and the software platform used in the experiments. In section 4, experimental results are presented showing the potential of the approach, followed by conclusions in section 5.

[1] In the context of genetic algorithms the evolutionary process means the application of the genetic operators: selection, crossover and mutation.

Algorithm:

```
initialization of the
population (M);
evaluatation (M);

while (true)
{
replication (M); ⇒ I
mutation (I); ⇒ I'
evaluation (I'); ⇒ I''
selection (I''); ⇒ M
}
end
```

Figure 1: Evolutionary algorithm.

2 EVOLUTIONARY ALGORITHMS

Evolutionary algorithms are inspired by principles of the evolution theory. The basic idea is to maintain a population of individuals (candidate solutions) which evolve under selective pressure that favours better solutions. The increasing interest in Evolutionary Algorithms is because their robust and powerful adaptive search mechanisms. For the interested reader see Fogel (1995).

In the following, a novel evolutionary algorithm which has similarities with evolution strategies and evolutionary programming (Fogel, 1995) is presented: The population is made up of two sub-populations: a memory population *M* and an innovation population *I*. The individuals of the innovation population undergo the operation of replication, mutation, and selection. The algorithm is shown in Fig.1.

Firstly, the memory population *M* of individuals represented as bit strings is randomly initialized, followed by the fitness evaluation. By applying the replication operator with size *N* to the memory population *M* results in the innovation population *I*, which after application of the operator mutation, results the population *I'*. The innovation population is evaluated and sorted according to the fitness in ascending order resulting in *I''*. The fittest individuals of *I''* compose then the memory population and the other individuals are discarded (die-out); this process represents one generation. The mutation process is carried out as follows: Individuals with high fitness have applied to them a low mutation rate, while individuals with low fitness are subjected to a high mutation rate. The tuning of the mutation rate depends on the problem, for the robot navigation a relation was defined for the mutation rate according to the fitness, which will be defined later. Recent work relating to adaptive mutation rates in genetic algorithms can be found in Thierens (2002) and references therein.

3 FPGA-BASED CONTROLLER

The implementation is based on intrinsic EHW which is evolved using a Xilinx Virtex FPGA. The genotype of each individual is mapped to the circuit on chip and the fitness is evaluated on-line. The architecture of the control system is depicted in Fig. 2. The following describes the Kephera robot, the hardware and the software used to carry out the experiments.

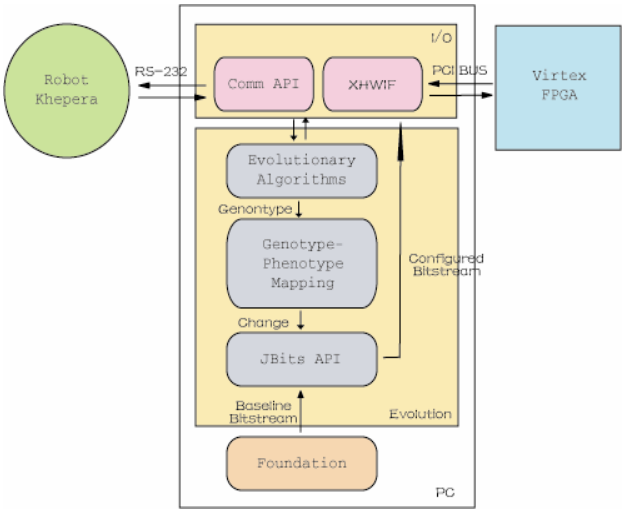


Figure 2: The control system architecture.

Figure 3: The Kephera robot.

3.1 THE KEPHERA ROBOT

The Khepera robot used in the experiments is shown in Fig. 3. It has eight infrared proximity sensors and 2 wheels (Michel, 1999). Each sensor can emit infrared light and detect the reflected signals. The sensor value is varied from 0-1023. The higher the sensor value the closer the distance between sensor and obstacle. A value of 400 is used here as a threshold value to convert eight sensor input into 8 bits of information. Each wheel of the robot is controlled by an independent DC motor. The controller receives 2 bit information for four commands: move forward, move backward, turn left and turn right. The robot is controlled by the host PC through the cable connecting between RS-232 and the robot.

3.2 THE HARDWARE

FPGA is a VLSI chip that comprises a matrix of configurable logic blocks (CLBs) surrounded by programmable input/output blocks (IOBs). With the routing and logic information stored in its memory block, the FPGA provides the desired function by the download of the configuration Bitstream. The FPGA used in this experiment is Xilinx's Virtex V1000, (Xilinx Datasheet, 2001) which is connected to a PC via a PCI interface card. In the past, most of the researchers used Xilinx XC6200 series in their experiments, because this design restricted the connection between the outputs of logic blocks in any random configuration, and hence the possibility of "dangerous" configurations. However, Xilinx discontinued to supply XC6200 and switched their product to Virtex series. Compared with the Xilinx XC6200, the Virtex device has multi-directional routing, which makes it possible to connect two outputs of logic gates together. This unsafe property prohibits researchers using random Bitstream to configure the routing connection of the device. One possible solution to this is to construct a XC6200 model on Virtex chip to restrict the routing (Hollingworth et al., 2000). This alternative way provides a safe method to implement both logic and routing modification on Virtex series FPGA. However, this solution has limits due to its simple routing model (Hollingworth, 2001). It is not easy for the input signal to pass through the matrix of Logic Cells. Also, the XC6200 model consumes considerable resources on the device. In order to evolve a safe and efficient controller in these experiments, only the logic function of the robot controller will be evolved, instead of both logic and routing. The robot controller is evolved using 22 LUTs (Look Up Tables) on the FPGA, with 8 input bits from the sensors, and 2 output bits to the motors as shown in Fig. 4.

The hardware interface is the XHWIF (Xilinx Hardware Interface), a java interface using native method to operate the hardware platform dependent part of the interface. In these experiments, it is used to communicate between FPGA and host PC.

3.3 THE SOFTWARE

JBits (Xilinx, 1999) is a set of JAVA classes which provides an Application Program Interface (API) into the Xilinx Virtex FPGA family Bitstream. In these experiments, this interface is used to operate on the Bitstream generated by Xilinx Foundation (a circuit design tool from Xilinx). It can dynamically modify the circuit design on Virtex implementing genotype mapping. The basic element that JBits operates on is the LUT. A LUT has four input vector and one output. There are $2^4 = 16$ information bits. For example, a true table of one four input OR gate is:

OR 1111 1111 1111 1110 0xfffe.

The following JBits source code is used to set one LUT to implement an OR gate.

JBits.set(1, 1, LUT.SLICE_G, 0xfffe).

Therefore, each controller consists of 22 LUTs * 16 bits/LUT = 352 bits. So, each individual is represented by 352 information bits.

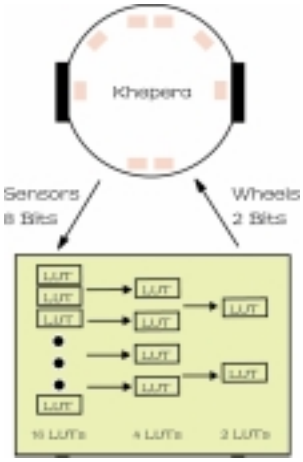


Figure 4: FPGA-based controller.

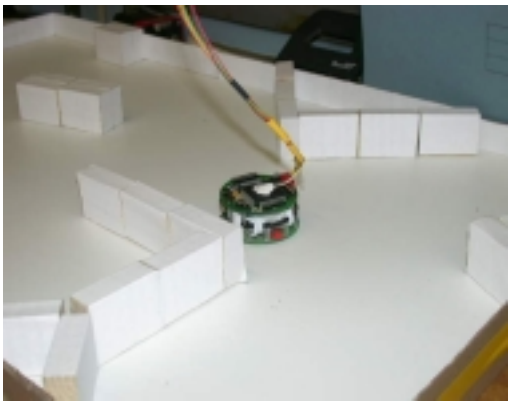


Figure 5: The experimental test environment for the Kephera robot navigation.

4 EXPERIMENTAL RESULTS

The experimental configuration used to test the algorithm is shown in Fig. 5.

4.1 THE FITNESS FUNCTION

The fitness measure is a very important part of an evolved robot controller. In on-line evolution, different individuals may not face the same environments. The wheels' speeds are determined by the real-time information from the sensors. The controller must send a sequence of different commands to the wheels in real-time. These make the fitness measure much more difficult to evaluate (Tan et al., 2002). To solve the problem, the fitness function used here will not try to compare the value between input information and output commands. It simply measures the time and distance the robot has run before it hits an obstacle, the longer the time and the longer the distance the higher the fitness value. So, the fitness value can be calculated according to a simple relation:

*fitness = distance * time/1000*

The *distance* is the value of the position counter of the two motors, and the *time* is the value of I/O period counter. The distance of about 1000 represents a movement of 40mm. If the robot is turning around, there will be no increment in distance value. In order to shorten the time, a time limit value is set to 140. An individual will be killed when the limit is reached; even if it has not hit the wall. At the same time, if the individual is stuck in some position without any distance improvement, it will be killed. The fitness limit in the experiment is 1400, which means the robot run forward without any steering change. In the experiments, the values of two nearby sensors are used to judge whether the robot is too close to the wall, and an escape time is provided to the individual to quit from the dead zone where the last one was killed.

4.2 PARAMETERS SETUP AND RESULTS

The evaluation of each individual on-line is very time consuming. Therefore, a small population size was used. For the experiments, the memory population has size 16, and the innovation population has $N = 3$ sets of 16 individuals each. So, the total population size amounts 64 (16 for the memory population + 3*16 for the innovation population). The memory population *M* is not mutated in order to make sure that good individuals of the memory *M* be not lost by the mutation operation.

In order to compare the results, a standard case was used, where all individuals of the innovation population have 8 bits per Bitstring mutated, i.e. a mutation rate equal 0.022. Fig. 6 shows the results for this case.

The mutation rate used, i.e., mutation tacking into account the fitness, is calculated according to the relation: *n**(1-*norm_fit*), where *n* stands for the number of bits and *norm_fit* designates the normalized fitness. Note that, in

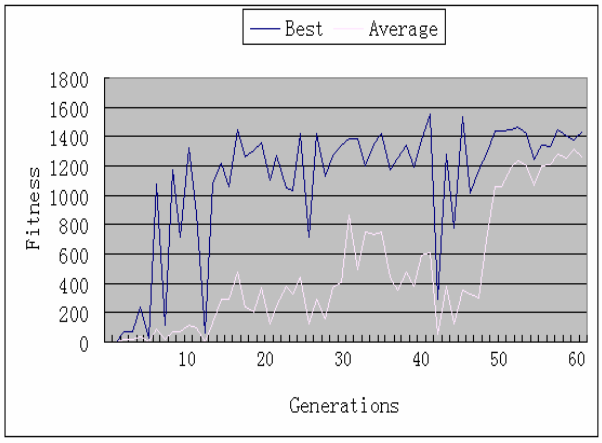


Figure 6: All individuals of the innovation population have equally 8 bits mutated.

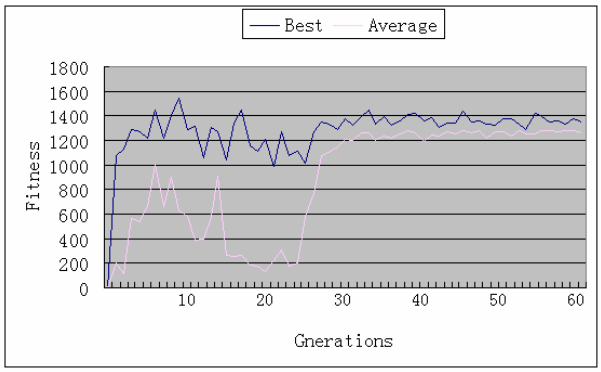


Figure 7: The mutation rate for each individual of the innovation population is calculated according to 16*(1-*norm_fit*). The best individual has no bits mutated and the worst individual has 16 bits mutated.

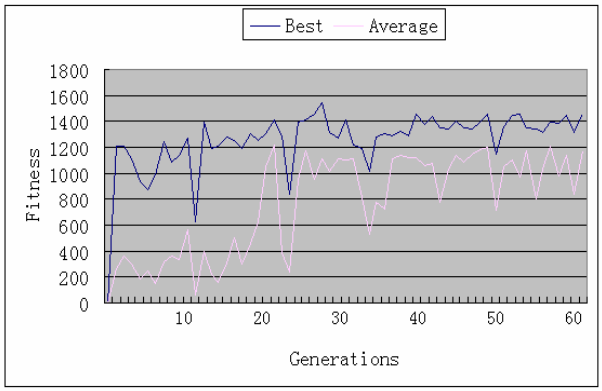


Figure 8: The mutation rate for each individual of the innovation population is calculated according to 35*(1-*norm_fit*). The best individual has no bits mutated and the worst individual has 35 bits mutated.

order to obtain an integer number of bits to be mutated, simply the integer part is taken. Two cases are considered: For n = 16, the best individual has normalized fitness 1, therefore no bits are mutated and the worst individual has normalized fitness 0, therefore 16 bits are mutated which corresponds to a mutation rate of 0.045. Fig. 7 shows the results for this case. In the next case studied, the mutation rate was increased. For n = 35, the worst individual had 35 bits mutated which corresponds a mutation rate of 0.1. The result for this third case is shown in Fig. 8.

4.3 DISCUSSIONS

The experimental results present some interesting features. In the first case as shown in Fig. 6, considering a constant mutation rate for all individuals of the innovation population, it can be noted that it is quite difficult to achieve the average fitness of 1200 which was reached after 50 generations. The fitness of the best individual presents an oscillating behaviour. In the second case as depicted in Fig. 7, the strategy examined consists of a mutation rate related to the fitness, whereas the number of mutated bits for the worst individual was 16. So, using adaptive mutation the average fitness reached the value 1200 in less than 30 generations with the fitness of the best individual presenting a quite stable behaviour around the maximum value of 1400. Therefore, capable to tracking quite well the dynamic environment. This case presents a substantial improvement in performance compared with the former case. In the third case studied shown in Fig. 8, the same mutation strategy was used but the mutation rate was increased, i.e., for the worst individual the number of mutated bits was 35. For this case, the behaviour of the average fitness is very oscillatory, but even so the fitness of the best individual presents a high value around 1400 with more fluctuation when compared with the former case, which is due the high mutation rate.

By means of the experiments carried out it can be observed that the mutation operator is a very important one to track dynamic environments as is the case of autonomous robot navigation. It might be argued that it is quite difficult to obtain adaptive behaviour with constant mutation rate. With an adaptive mutation rate it was possible to obtain good results with an intermediary mutation rate, which is dependent of the problem on hand. Also, the results demonstrated that a too high mutation rate might lead to an unstable behaviour, i.e., the inability to track change of the environment. In (Travis and Travis, 2002) it was pointed out the role of mutator clones in adaptation of organisms to fluctuating environments. In their experiments with different mutation rates they noticed the influence of different mutation rate leading to different behaviours, inclusive chaos at high mutation rate. It may be that biological inspiration provides us with new insights to evolve adaptive behaviour to dynamic / unknown environments.

5 CONCLUSIONS

This paper presented a novel evolutionary algorithm to evolving FPGA-based controller for autonomous navigation of a mobile robot. The algorithm, in which the mutation rate is defined according to the normalized fitness has demonstrated suitability to tracking dynamic environment. Experimental results on the Kephera robot have shown that the algorithm is capable of providing autonomous navigation in real time for collision avoidance. In particular, the new operator mutation proposed, has been examined for different mutation rates. It was observed that different behaviours emerge, e.g. good tracking of dynamic changes at intermediary mutation rates, and the presence of fluctuations, or instability, for high mutation rate.

Since the presented algorithm is simpler than genetic algorithms because no crossover is used, its implementation into an FPGA allowing the whole evolutionary process execution in hardware is highly desirable. Further work includes investigating the performance of the algorithm for fault tolerance.

Acknowledgements

The authors would like to thank Dr. G. Hollingworth and Dr. R. Canham for providing the hardware setup and basic source code in Java.

"This project is funded by the Future and Emerging Technologies programme (IST-FET) for the European Community, under grant IST-2000-28027 (POETIC). The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication."

References

D.B. Fogel (1995). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, Piscataway.

P. Haddow, and G. Tufte (1999). Evolving a robot controller in hardware, *In Proc. of the Norwegian Computer Science Conference* (NIK-99), pp. 141-150.

P. Haddow, and G. Tufte (2000). An evolvable hardware FPGA for adaptive hardware. *In Congress on Evolutionary Computation (CEC00)*, pp. 553-560.

G.S. Hollingworth (2001). *Fault tolerance of evolvable hardware through diversification*, Ph.D. Thesis, The University of York, UK.

G. Hollingworth, S. Smith, and A.M. Tyrrell (2000). Safe intrinsic evolution of virtex devices. *In the second NASA/DoD Workshop on Evolvable Hardware*, IEEE Press, pp. 195-202.

D. Keymeulen, K. Konaka, M. Iwata, et al. (1997). Robot learning using gate level evolvable hardware, *In Proc. of*

the Sixth European Workshop on Learning Robots (EWLR-6), Springer-Verlag.

P. Layzell (1998). A new research tool for intrinsic hardware evolution. *In second International Conference on Evolvable Systems* (ICES-98), Lecture Notes in Computer Science, Springer-Verlag, pp. 47-56.

O. Michel (1999). *Khepera User Manual. K-TEAM*, Version 5.02.

J. Miller, and P. Thomson (1998) Aspects of digital evolution: Evolvability and architecture. *In Fifth Intern. Conf. on Parallel Problem Solving from Nature* (PPSN-98), Springer-Verlag.

S. Shirasuchi (1996). FPGA as a key component for reconfigurable system, *In First International Conference on Evolvable Systems* (ICES-96), Lecture Notes in Computer Science, Springer-Verlag, pp. 23-32.

K.C.Tan, C.M. Chew, K.K. Tan, L.F. Wang, and Y. J. Chen (2002). Autonomous robot navigation via intrinsic evolution. *In Proc. of the congress on evolutionary computation, part of the IEEE world congress on computational intelligence*, Honolulu, USA, pp.1272-1277.

D. Thierens (2002). Adaptive mutation rate control schemes in genetic algorithms. *In Proc. of the congress on evolutionary computation, part of the IEEE world congress on computational intelligence*, Honolulu, USA, pp. 980-985.

A. Thompson (1995). Evolving electronic robot controllers that exploit hardware resources. *In Proc. of the third European Conf. on Artificial Life* (ECAL-95), Springer-Verlag, pp. 640-656.

A. Thompson (1996). An evolved circuit, intrinsic in silicon, entwined with physics. *In First International Conference on Evolvable Systems* (ICES-96), Lecture Notes in Computer Science, Springer-Verlag, pp. 390-405.

Xilinx, (1999). *Xilinx JBits documentation*. Published in JBits. Version 2.0.1.

Xilinx datasheet, (2001). Xilinx Inc. *Virtex Field Programmable Gate Arrays data book*. Version 2.5.

http://www.xilinx.com/partinfo/ds003-1.pdf

R.S. Zebulum, M.M.R. Vellasco, and M.A.C. Pacheco (2001). *Evolutionary electronics: Automatic design of electronic circuits*. CRC Press, Boca Raton, FL, USA.

Exploiting the analogy between immunology and sparse distributed memories: a system for clustering non-stationary data

Emma Hart and Peter Ross,
Napier University, Scotland
{e.hart,p.ross}@napier.ac.uk

Abstract

The relationship between immunological memory and a class of associative memories known as sparse distributed memories (SDM) is well known. This paper proposes a new model for clustering non-stationary data based on a combination of salient features from the two metaphors. The resulting system embodies the important principles of both types of memory; it is self-organising, robust, scalable, dynamic and can perform anomaly detection. The model is first applied to clustering static datasets, and is shown to outperform two other systems based on immunological principles. It is then applied to clustering non-stationary data-sets with promising results.

1 INTRODUCTION

Modern technology makes it incredibly straightforward for companies to gather vast amounts of data concerning individuals and their habits on a daily basis, for example through the use of credit cards or supermarket loyalty cards. Interpreting such huge quantities of data, and identifying clusters and trends within it is a mammoth task, especially as the data may be rapidly changing. Data-clustering can be defined as *"the unsupervised classification of patterns (observations, data items or feature vectors) into groups (clusters)"* [Jain et al., 1999], and is performed in the hope that implicit previously unknown and potentially useful knowledge can be extracted from the data. It is a large field in its own right, and there are many documented approaches.

However, the immune metaphor may provide a novel and alternative approach. Both the immune system and a data-clustering system have to operate in very large input spaces. In the immune system, a lymphocyte recognises a *set* of antigens, due to its imprecise matching characteristics; that set can be considered to be equivalent to a cluster within a database. The lymphocyte that recognises all the items in a cluster thus provides a concise description of the cluster itself. The number of lymphocytes present and the specificity of the recognition process provides a mechanism for controlling the number of clusters present, and hence provides a method of controlling how specifically (or generally) the clusters are described. The fact that recognition is imprecise is important — data in a database is likely to contain much noise and redundant information, therefore some kind of imprecise recognition mechanism will be essential.

The natural immune system can react to unseen pathogens either by producing new lymphocytes using its inbuilt diversity generating mechanisms or by adapting existing lymphocytes via mutation mechanisms. Similarly, when new data arrives in the database, the centres and sizes of the clusters may need to move and adapt in order to recognise the new data. New cluster centres may be created and old ones may disappear over the course of time, the key point being that the system can respond dynamically to the state of the database at a given moment in time.

The natural immune system is very efficient at recognising the sudden appearance of harmful invaders; a data-clustering system should be able to recognise the appearance of anomalous data in the database. This feature would automatically result from an immune based model however — if a data-item is not recognised by any cluster it would result in the system having to create a new lymphocyte. External observation that this has occurred would signify that perhaps that the data is non-representative of the general patterns and therefore triggering some warning. Imagine for example attempting to cluster data collected by a credit-

card company relating to card usage. The company is interested in clustering the data to identify patterns in card usage, but would also like to detect fraudulent card-usage. If a newly presented data-item does not belong to an already established cluster, it could identify an attempt at fraudulent usage of the card, which further human examination could verify. Finally, the distributed nature of the immune system architecture is attractive, given the fact that very large datasets are also likely to be distributed.

2 Related Work

Several of the features just described have been modelled in a number of very different implementations of artificial immune systems and applied to the problem of clustering data. For example, Potter *et al.* describe a model of an AIS that uses a coevolutionary genetic algorithm (GA) to evolve antibodies to cluster artificial data sets [Potter and De Jong, 2000] and Congress voting records [Potter and De Jong, 1998]; Forrest *et. al* [Forrest et al., 1993] describe a GA that uses emergent fitness sharing to find patterns; Hunt *et al.* [Hunt et al., 1999] describe a system named *Jisys* which was used to cluster data for use in mortgage fraud detection and Timmis [Timmis et al., 2000, Timmis and Neal, 2001] has adapted this system to successfully cluster the well known but very small benchmark data set containing iris petal sizes. Both the Timmis and Hunt work used a model based on connected networks of antibodies, in which nodes which are connected recognise similar patterns. A similar approach was adopted by [De Castro and Von Zuben, 2000] who present a network model for data clustering and filtering redundant data. So far, none of these methods have addressed the question of clustering data in time-varying databases. Although there is no intrinsic barrier to extending either the coevolutionary or network models to deal with non-stationary data, both methods present obstacles. In the network model, there are high computational overheads associated with re-organising large networks as the data changes, which increase as the size of the database increases also. It is also unclear whether the coevolutionary method of evolving clusters is able to cope with extremely large databases, particularly as the antibodies compete to exclusively recognise data, whereas in reality clusters may overlap.

3 Exploiting the correspondence between immunology and SDM

Smith *et. al* [Smith et al., 1999] have shown that the immune system can be considered to be representative of the same class of memories as Kanerva's Sparse Distributed memory, [Kanerva, 1988]. The SDM is a content-addressable memory which was originally proposed as an efficient method for storing a very large number of large binary data patterns using a very small number of physical data addresses, in a manner which allows accurate recall of all the data. An SDM is composed of a set of physical or hard locations, each of which recognises data within a specified distance of itself — this distance is known as the recognition radius of the location. Each location also has an associated set of counters, one for each bit in its length, which it uses to 'vote' on whether a bit recalled from the memory should be set to 1 or 0. An item of data is stored in the memory by distributing it to every location which recognises it — if recognition occurs, then the counters at the recognising locations are updated by either incrementing the counter by 1 if the bit being stored is 1, or decrementing the counter by 1 if the bit being stored is 0. To recall data from the memory, all locations which recognise an address from which recall is being attempted vote by summing their counters at each bit position; a positive sum results in the recalled bit being set to 1, a negative sum in the bit being set to 0. This results in a memory which is particularly robust to noisy data due to its distributed nature and inexact method of storing data. These properties make it an ideal candidate for addressing clustering problems in large databases. For example, we can consider each physical location along with its recognition radius to define a cluster of data; the location itself can be considered to be a concise representation or description of that cluster, and the recognition radius specifies the size of the cluster. Clusters can overlap — indeed, it is this precisely this property which allows all data to be recognised with high precision whilst maintaining a relatively low number of clusters. If no overlap is allowed, then a large number of locations are required to cluster the data, the system becomes overly specific, and hence general trends in the data are lost. In the form described, the SDM is also static and inflexible, however given its powerful and efficient storage and recognition capacities, it is fruitful to adapt it to operate in a dynamic environment.

Previous work by the authors [Hart and Ross, 2001] presented an immune system model for clustering moving datasets based on an SDM that was dynamic, adaptable and capable of tracking changes in large vol-

umes of data. In this model, an antigen is equivalent to a piece of data, an antibody to a description of a cluster, and the ball of stimulation of the antibody defines the size of the cluster. The basic proposition of the model was to use a coevolutionary GA, running continuously, to find quickly the set of antibodies (and their corresponding balls of stimulation) that best cluster the data currently visible to the system. Whilst some sucesss was observed, the model suffered from three drawbacks; namely, that the evolved SDMs failed to recognise some antigen altogether, that correctly setting the recognition radii of each centre is extremely difficult, and that the algorithm is relatively slow, due to the nature of the fitness function. This paper presents a new model, the self-organising SDM or SOSDM, which closely models the self-organising nature of the biological immune system, one of its fundamental characteristics. SOSDM views immunological memory as a truly self-organising system. Initially randomly placed hard locations self-organise in order that they become distributed throughout the input data space in a manner which reflects the input data distribution. This seems an entirely logical step — the immune system itself is self-organising, whilst viewed from the computational angle, there is an abundance of literature describing algorithms for self-organising systems, the classic example of course being the Kohonen network, [Kohonen, 1982]. Furthermore, a number of data-clustering algorithms rely on self-organising principles, and the existing immune-network models that perform data clustering are also self-organising,[De Castro and Von Zuben, 2000, Timmis et al., 2000] and [Timmis et al., 1999] compares an AIS to a Kohonen network.

Despite its similarity to immunological memory, the SDM in its original form is unsuitable for clustering data, for reasons described in detail in [Hart and Ross, 2001]. However, [Hely et al., 1997] have proposed an alternative model of an SDM which was developed in order to handle non-random input data more satisfactorily then Kanerva's original system. According to [Hely et al., 1997],"*the SDM signal model retains the essential characteristics of the original SDM whilst providing the memory with a greater scope for plasticity and self-evolution. By removing many of the problematic features of the original SDM the new model is not as dependent upon* a priori *input values.*". In Hely's model, the storage locations that make up the final memory are *not* known from the start. Initially locations are created until there is an excess of storage locations which then compete for available signal (i.e. data). Storage locations receiving little or no signal are removed. Locations which survive are chosen for the total amount of signal they receive. The recognition radius of the original SDM is replaced by a new parameter which decreases the value of the signal as it spreads out. Locations have real valued counters to store a copy of the data weighted by the strength of signal they receive. The signal does not propagate after it falls below a minimum strength. Importantly, in the context of data-clustering, this model does not rely on a single parameter defining a recognition radius, and furthermore does not depend on locations being randomly distributed throughout the input space; clearly, the input data in a database is not random. Our new model SOSDM, borrows from the underlying philosophy of the Hely signal model of distributing data, but modifies the detail somewhat. Data is distributed to many locations with decreasing strength, but we also take inspiration from the algorithm used by [Potter and De Jong, 2000] in that centres *compete* for data based on their *affinity* for the data. In order cope with the demands of clustering non-sationary data which requires the memory to be flexible in terms of the number of centres present at any given time, nodes are added and deleted only as necessary in areas of the input space that are misrepresented given the current state of the data.

4 Implementation of SOSDM

Pseudo-code outling the SOSDM algorithm is given in figure 1. The basic principles are as follows: Firstly, input data patterns (or a random subset of the dataset) are distributed to a subset of the hard locations, based on the *affinity* of each hard location for the data-item in a batch process. This results in the counters of the subset of locations being updated, according to the strength of each signal received. After all signals have been propagated, the accumulated error at each location is calculated. The error is equivalent to the sum of the *distances* between each node and any data it recognises, weighted by the signal strength. The value of the error is then used to allow the hard locations to self-organise — locations gravitate towards areas of the space in which they recognise data, the distance and direction of the movement determined by the accumulated error. Each of these steps is now described in greater detail.

4.1 Distributing the Data

Data is distributed through the SDM according to the *affinity* $\mathcal{A}$ of each centre c for an input signal a. This is defined simply as Hamming Distance between the input data and the *address* of the centre c (equation 1).

1. begin with a fixed number of centres N, with randomly initialised positions and counters set to 0.
2. present a subset $s(s \leq N)$ of the data-set visible at time t to the SOSDM
3. distribute the data in the s to *each centre* in the SOSDM, with a *strength* proportional to the *affinity* of the centre for the data
   - update the counters at each centre according to the strength of signal received
   - compute the accumulated error at each centre
4. update centre positions — the distance and direction of the move is determined by the total accumulated error at the centre
5. update centre counters
6. add or delete nodes from the memory if necessary
7. go back to step 2

Figure 1: The SOSDM algorithm

$$\mathcal{A}(c_i, a) = \sum_{j=1}^{j=L} \begin{cases} 1 & if \quad V(c_j) = a_j \\ 0 & otherwise \end{cases} \tag{1}$$

The affinity of each of the N centres for the input data is calculated. Following this, the centre with the highest affinity for the antigen a, denoted by $\mathcal{A}^*$ can be determined:

$$\mathcal{A}^* = max(\mathcal{A}(c_1, a), ..., \mathcal{A}(c_N, a)) \tag{2}$$

This value $\mathcal{A}^*$ is then used to determine how much of a signal is distributed to each centre. Signal is distributed according to its *strength*, where the strength of a signal at centre c_i is proportional to the ratio of the affinity of that centre for the signal, $\mathcal{A}(c_i, a)$ to $\mathcal{A}^*$. A further parameter known as the signal-threshold t is introduced, such that $(0 \leq t \leq 1)$. Signal is only distributed to those centres in which the strength of signal is greater than this threshold. This is shown in equation 3.

$$\mathcal{S}(c, a) = \begin{cases} \frac{\mathcal{A}}{\mathcal{A}^*} & if \quad \mathcal{S}(c, a) > t \\ 0 & otherwise \end{cases} \tag{3}$$

Distributing a signal to a centre implies updating the counters at that centre. The counter $\mathcal{C}(c_{ij})$ for each bit j at each centre c_i is updated according to equation 4, where γ is equal to 1 if $V(c_{ij})1$ and to -1 if $V(c_{ij}) = 0$.

$$\mathcal{C}_(c_{ij}) = \mathcal{C}(c_{ij}) \ + \gamma \mathcal{S}(c, a) \tag{4}$$

As signal is distributed in this manner, a running total of the accumulated signal received at each centre is incremented, as shown in equation 5:

$$\forall c: \quad \mathcal{R}(c) = \mathcal{R}(c) \ + \mathcal{S}(c, a) \tag{5}$$

4.2 Calculating the Error at Each Node

The self-organising mechanism by which centres move around the SDM is based on a calculation of the total error accumulated at each centre *after all* input signals have been distributed to the SDM. Error is calculated in the following manner; firstly, each time a signal is stored at some centre c, the error *at each of the j bit positions* for the address of that centre is updated according to equation 6. The error at each bit position is this effectively a measure of the difference between the *desired* value of the centre address at position j as given by the value of the antigen at position j and the *actual* value of the centre address, $V(c_{ij})$.

$$\mathcal{E}(c_{ij}) = \mathcal{E}(c_{ij}) \ + \mathcal{S}(c_i, a)(a_j - V(c_{ij})) \tag{6}$$

Movement of centres only occurs after all data has been presented to the SDM, which allows the total average error at each centre, $\overline{\mathcal{E}}$, to be calculated, according to equation 7. Note that this will always have

a value lying between -1 and 1.

$$\overline{\mathcal{E}}(c_{ij}) = \mathcal{E}(c_{ij})/\mathcal{R}(c_i) \quad (7)$$

4.3 Updating the nodes position and counters

Once all data has been presented, self-organisation of the centres can take place. Thus, as shown identified in steps 4 and 5 of the SOSDM algorithm in figure 1, the *address* of each centre is modified as the centres move to parts of the input space more representative of the signal they are receiving. The counters associated with a centre also move, however they too are modified as the physical locations of the centres move to reflect the new position of the centre.

The probability with which the position *and* the counter of each bit j in a centre c_i are moved is defined according to the *absolute* value of the average error $\overline{\mathcal{E}}(c_{ij})$. If the value of $|\overline{\mathcal{E}}(c_{ij})|$ is greater than 0.5, then this value determines the probability with which an address bit is flipped and its counter updated. (The introduction of the value of 0.5 ensures that the system will eventually stabilize given a static data set and prevents random movements). Thus, if $\overline{\mathcal{E}}(c_{ij}) < 0$, then $V(c_{ij}) \Rightarrow 0$, and if $\overline{\mathcal{E}}(c_{ij}) > 0$, then $V(c_{ij}) \Rightarrow 1.$. Equation 8 summarises the effect on the *counters* for each bit j in each centre c_i for all centres in which $|\overline{\mathcal{E}}(c_{ij})| > 0.5$. A new parameter is introduced — the *influence-counter*, $\mathcal{I}$. This parameter allows the amount by which the counters are adjusted to be explicitly controlled.

$$C(c_{ij}) \Rightarrow C(c_{ij}) \times \left(1 + \left(\mathcal{I} \times \overline{\mathcal{E}}(c_{ij})\right)\right) \quad (8)$$

Thus, the effect on a counter is that it is increased or decreased by a percentage of its original value, the amount of which is proportional to the total error accumulated at the node. The effect on the address of bit j is that it is flipped, with a probability proportional to the average error accumulated at that address location.

In summary, the key features of the SOSDM algorithm involve distributing a sample of data to the system, followed by allowing the system to self-organise, in a manner dependent on the average error accumulated at each centre. The algorithm is iterated until it stabilises (given a static data set). Note that when using SOSDM there is no need to calculate the mean recall accuracy of the system at each iteration, as with COSDM. The value of this parameter does not feedback into the algorithm and has no bearing on its performance. However, in order for the observer to evaluate the performance of SOSDM, this quantity must be calculated. The method by which this is done is now outlined.

4.4 Recalling Data from the SOSDM

The quality of the SOSDM defined by this model is measured by the accuracy with which data stored in the memory can subsequently be recalled.

When attempting to recall an antigen a, first the antigen that is retrieved from the memory a' is calculated, and then this is compared to the desired antigen, i.e. that which was originally stored in the memory, a. The process is as follows:

- Calculate the subset of centres n' for which the signal strength $\mathcal{S}(c_i, a) > t$
- Sum the counters of each member of the subset n' at each of the j bit positions to give $\sigma_j(a)$. The value of each counter $C(c_{ij})$ is weighted by the strength $\mathcal{S}(c_i, a)$ of the signal during the summation process, as shown in equation 9.

$$\sigma_j = \sum_{i \in n'} C(c_{ij}, a) \times \mathcal{S}(c_i, a) \quad (9)$$

Then, as in the original SDM, any bit where $\sigma_j > 0$ has a 1 at that location in the recalled data, and any bit where $sigma_j < 0$ has a 0 at that location in the recalled data. Thus, the actual recalled antigen is calculated, compared to the desired antigen, and the match-score $\mathcal{M}$ between the actual and desired antigen derived. This is simply the number of bit positions in which the recalled data and original data have identical values. The match-score is then used to calculate the mean recall accuracy, $\overline{r}$, which is a quantitative measure of the performance of the system, and is given in equation 10.

$$Mean\ recall\ accuracy\ \overline{r} = \frac{1}{N} \sum_{i=1}^{i=N} \mathcal{M}(a'_i, a_i) \quad (10)$$

This quantity thus measures the number of matching bits between a recalled antigen and the original stored antigen, and hence has a value between 0 and L, where L is the length of the antigen.

4.5 Experimental Set-up

Experiments were first performed using static datasets, using the same binary data-sets described

by Potter in [Potter and De Jong, 2000] and in [Hart and Ross, 2001]. There are 3 datasets, the first containing 2 clusters, the second 4 clusters, and the third 8 clusters, identified as half-schemas, quarter-schemas, and eighth-schemas respectively. The half-schema set is generated in equal proportion from 2 schemas — in schema-1, the first $L/2$ bits of the schema are set to 1, the remainder to wild-cards, in schema-2, the latter $L/2$ bits are set to 1, with the first $L/2$ bits set to wild-cards. Similary, quarter-schema data-sets are generated in equal proportion from 4 such schemas, and eighth-schema data-sets from 8 such schemas. In each case, the results can be compared against the match that would be obtained by the best possible single string generalist, which would always consist of a string containing all 1's, thus resulting in a match-score of $(L/x) + (\frac{L-(L/x)}{2})$, where x is 2,4 or 8 when using half, quarter or eighth schema datasets respectively. The number of antigen in each dataset is varied from 5 to 500, in steps of 50, and the length of each antigen string in each case is 64. Unless stated otherwise, each experiment is repeated 10 times, and the SOSDM algorithm is applied for 200 iterations. The quality of the algorithm is measured by the mean recalled accuracy, (see equation 10). The number of centres in each experiment was fixed before the experiment began, and remained static throughout each experiment, as the number of clusters in each dataset is known *a priori*. Results are compared with those of the co-evolutionary immune algorithm given in [Potter and De Jong, 2000] — this system is referred to as CE-POTTER in the remainder of this paper.

4.6 Comparison of SOSDM Performance to that of CE-POTTER

Initial experiments were performed with $t = 1.0$ and $\mathcal{I} = 1.0$. Thus, data is distributed to *all* centres with $\mathcal{A} = \mathcal{A}^*$ and to no others. (This is in direct comparison to the Potter approach in which data is only distributed to a single centre, with ties broken by age of centre). The setting for I also ensures that counters are adjusted maximally. The best recall-accuracy obtained in each of 10 experiments is recorded, and the results averaged. Figure 2 shows a plot of the results — clearly SOSDM outperforms CE-POTTER for all sizes of antigen dataset and regardless of the number of clusters. T-tests show that the mean recalled accuracy obtained using SOSDM is statistically significant in every case when compared to the identical experiment using CE-POTTER.

Examining the results in more detail shows that as the number of antigen increases, the number of centres receiving the maximum strength of a data-item increases in all experiments. In small datasets, it is relatively straightforward for the centres to distinguish between each cluster. For very large datasets however, even though the data items nominally belong to separate clusters, there is likely to be a large overlap between items in each cluster, especially as the length of the defined section characterising each cluster decreases, and the number of antigens generated from that schema increases. Thus, the memory must generalise in order to accurately recall the large number of data-item, despite the fact that items nominally belong to a finite set of clusters — this is achieved by allowing clusters to overlap. This effect is much more clearly apparent in the quarter-schema and eighth-schema than it is for those using half-schema.

4.7 Performance vs Length of Antigen

A second series of experiments used datasets generated from quarter-schema, this time of fixed size N=200 antigens. The length of the antigen L in each dataset was varied from 40 to 1000 in steps of 40. The best recalled accuracy $\overline{r}$ was measured at the end of 200 iterations of SOSDM, and the results averaged over 50 trials. Figure 3 shows the results of these experiments; a comparison is made to the mean recall accuracy that would be expected using the best possible single string generalist for each value of L. The figure shows a direct correspondence between $\overline{r}$ and L — again, for every value of L , the value of $\overline{r}$ exceeds that expected using the single string generalist and this difference increases as L increases.

4.8 Performance of SOSDM vs Size of Dataset

Experiments were performed using datasets ranging in size from 500 antigen to 10,000 antigen in steps of 500. All datasets were generated using 4 quarter-schema, and the mean recalled accuracy of the entire dataset measured at the end of 200 iterations of the algorithm. Experiments were repeated 50 times in each case. Figure 4 shows the performance of SOSDM vs the size of of the dataset, with errorbars showing the minimum and maximum accuracy over the 50 experiments. Note that although there is a slight downwards trend in mean recalled accuracy $\overline{r}$ the value of $\overline{r}$ is always significantly greater than the result that would be obtained using the best possible string generalist, which would give $\overline{r} = 40$. T-tests show what there is a significant difference ($p > 0.99$) in the value of $\overline{r}$ obtained for $N = 1000$ and that obtained when

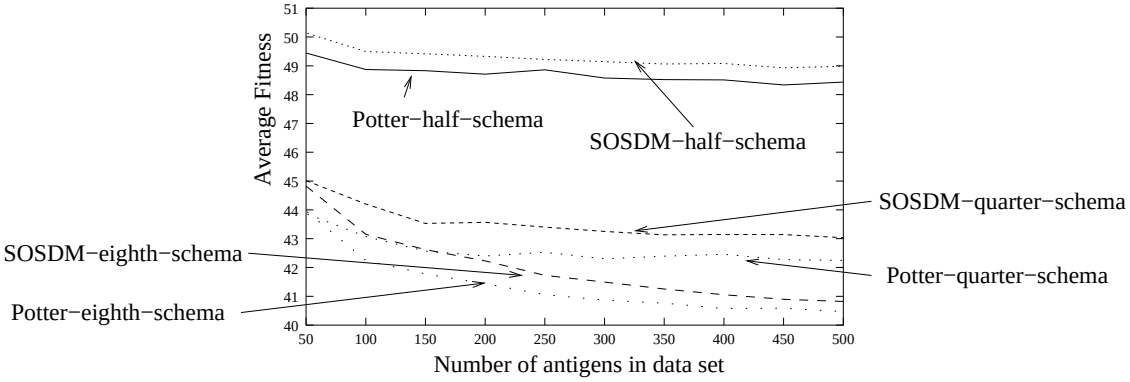


Figure 2: Comparison of Potter Algorithm to SOSDM for all experiments

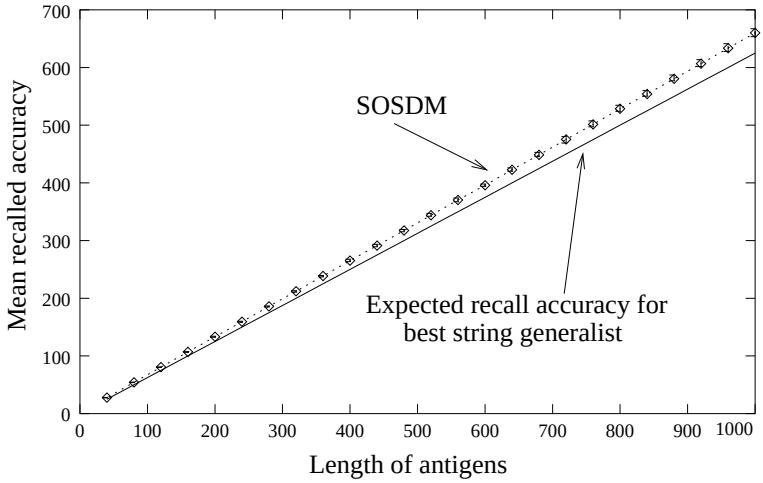


Figure 3: The figure shows how mean recalled accuracy $\overline{r}$ varies with the length of antigen L in a dataset of fixed size 200

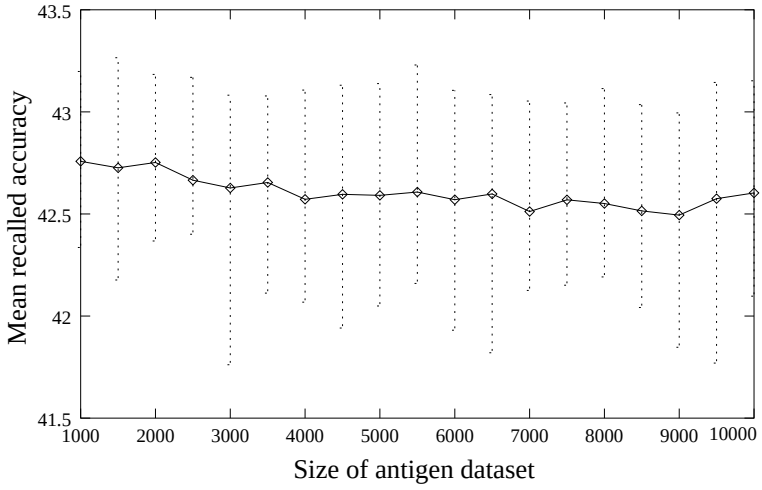


Figure 4: The figure shows how mean recalled accuracy $\overline{r}$ varies with the size of the antigen dataset N

Data	No. Centres (original data)	Average No. Centres using SOSDM	Average Recall Accuracy
half-schema	2	2.29	49.41
quarter-schema	4	6.75	44.77
eighth-schema	8	10.06	42.40

Table 1: The table shows the average number of centres required and corresponding accuracy of recall for clustering data-sets with a dynamic SOSDM algorithm

$N = 10,000$.

5 Non-stationary Data

In order for SOSDM to operate in a truly unsupervised manner in a non-stationary environment, SOSDM should be able to create and delete centres in response to the data it is exposed to, as generally the number of centres required will not be known *a priori*. The model must also contain a mechanism for both adding and removing centres as appropriate, depedning on the current state of the environment. In order to add centres to the model, a mechanism is suggested in which stagnation of the system is is detected not in respect to recall accuracy but in terms of movement of centres — if no movement of any centre has happened over a fixed number of generations s (the stagnation threshold) then a centre is added. The new centre is generated in a random position with its counters initialised to zeros. To delete centres that have become obselete, the total strength of signal $\mathcal{S}$ received by a node is compared to the total signal that the node has been exposed to; if $\mathcal{S}$ is less than some predefined percentage d (the deletion threshold), then the centre is deleted. However, a centre is allowed to exist for at least n epochs after creation in order to give it an opportunity to survive. Furthermore, a caveat is applied that if a centre uniquely recognises at least one antigen, then it is allowed to remain.

6 Results

A series of experiments was performed (see [Hart, 2002] for details) in which SOSDM was used to try and cluster the half-schema, quarter-schema and eighth-schema data used throughout this thesis. Each dataset contained 200 antigens, and in each experiment SOSDM was initialised with 2 centres. The stagnation threshold s is set to 10 iterations, and the deletion threshold d was varied as described below. At the end of each experiment, the best recall accuracy and the corresponding number of centres in the system are recorded. Each experiment was run 100 times and the results averaged. Initial experiments using the half-schema data showed that the actual value of the deletion threshold parameter d was unimportant in terms of the recall accuracy the system achieved and the average number of centres used, however it had a large effect on the number of times centres were deleted from the system and then subsequently re-added, hence a careful choice is necessary in order to make the system efficient. Results to be reported elsewhere clearly indicate that for this data, a large increase in the instability of the system occurs when the deletion threshold rises above 0.3. However, for all values of d, the system always produces its best results when the number of centres is on average 2, as desired. Experiments with the quarter-schema data and eighth-schema data were performed with d set to 0.25. The average number of centres required to give the best recall is shown in table 1.

The number of clusters in each case is sensible — although the original data-sets were created using 2,4 and 8 schemas and hence nominally contain the corresponding number of clusters, these clusters are somewhat arbitrary. Recall that the data is created by randomly filling in wild-cards in a set of schemas, therefore the formation of other clusters is likely, especially when the defined length of the schemas is short. Thus, with the half-schema data, the data is most accurately recalled using 2 or 3 clusters, closely matching the original schemas, whereas in the eighth-schema data, more accurate recall is gained by using more than the 8 clusters that the data was generated from.

7 Results with Non-Stationary Datasets

The experiments described in [Hart and Ross, 2001] using the coevolutionary immune model were repeated with SOSDM. The experiments examined the performance of SOSDM on a series of datasets in which new clusters are introduced at regular time-intervals, replacing random clusters in the original data-sets, using the following procedure:

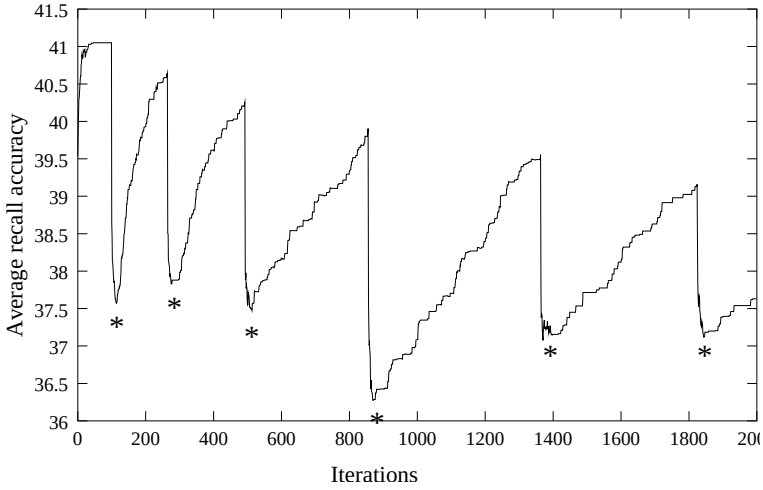


Figure 5: The figure shows how the recall accuracy of the SOSDM changes following a number of antigen updates in which entirely new clusters are introduced. The points marked * indicate the iteration at which the antigens were updated. The data contains 5 clusters, each containing 40 antigens. One cluster is replaced at each update.

The system is continually exposed to a set of 100 antigen. The antigen are generated from s schemas. Each schema consists of a string of 64 bits, in which c contiguous bits are set to 1, with the start position of the c bits randomly chosen. All remaining bit positions contain wild-cards. Antigen are generated in equal proportion from each schema by randomly replacing wild-cards with either 0 or 1. In order to generate non-stationary data, the following procedure is followed. 100 antigens are generated at time $t = 0$ from s schema. Every U time-steps, g schemas are chosen at random and replaced by g new randomly generated schema. New antigens are generated from the new schema and replace those antigens generated from the schema being replaced.

9 sets of experiments were performed, in which data sets were generated using $2, 5, 10$ schemas of length 64 bits, and the defined section of each schema was set to either 8,16 or 32 bits. For each dataset containing c clusters, experiments tested the ability of SOSDM to respond to replacing $1, 2, .., c$ clusters at each update, resulting in a total of 51 experiments. Figure 5 shows a typical result of one of the experiments in which the dataset was generated from 5 schemas each with 8 defined bits, and in which cluster was replaced at each update. It is difficult to observe clear trends in the results by varying either the number of clusters in the dataset or the number of defined bits in a cluster. However, it is observed that in all but one of the experiments, the average time lag for the system to return to its previous best level of fitness increases as the number of updates in one experiment increases. Also, it becomes increasingly difficult for the system to respond as the number of clusters being replaced increases. However, these experiments represent an extreme test of the system — in real life, entire new clusters are unlikely to suddenly appear at the same time as other clusters suddenly disappear, rather a more gradual process would occur. Methods by which the model could be improved in this respect are currently being investigated.

8 Conclusion

This paper has presented a new model for clustering both static and non-static data that is based on a combination of the ideas from both immunology and sparse distributed memories. The model outperforms previously published work on static data-sets, and furthermore, the results are shown to be scalable with the size of the data-set and with the length of the antigen data. An appealing feature of the model is the small number of parameters to be set — there are just two parameters and as yet unpublished work has shown that finding suitable settings for them is trivial. When tested with non-stationary datasets, good peformance is observed — although performance degrades as the number of clusters replaced increases, the mean recall accuracy of data from the clusters remains high. Given the short computational times required to achieve results, a possible solution to this problem would be to simply restart the system from scratch periodically. However, methods to control the rate at which the system can 'forget' data, more consistent with the im-

mune metaphor are currently being investigated.

References

[De Castro and Von Zuben, 2000] De Castro, L. and Von Zuben, F. (2000). An evolutionary immune network for data clustering. In *Proceedings of the IEEE Brazilian Symposium on Artificial Neural Networks*, pages 84–89.

[Forrest et al., 1993] Forrest, S., Javornik, B., Smith, R., and Perelson, A. (1993). Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation*, 1(3):191–211.

[Hart, 2002] Hart, E. (2002). Immunology as a metaphor for computational information processing: Fact or fiction. PhD thesis, University of Edinburgh, submitted July 2002.

[Hart and Ross, 2001] Hart, E. and Ross, P. (2001). Clustering moving data with a modified immune algorithm. In Boers, E. *et al.*, editor, *Applications of Evolutionary Computing, EvoWorkshops 2001*, number 2037 in LNCS, pages 394–404. Springer.

[Hely et al., 1997] Hely, T., Willshaw, D., and Hayes, G. (1997). A new approach to kanerva's sparse distributed memory. In *IEEE Transactions on Neural Networks*, pages 101–105.

[Hunt et al., 1999] Hunt, J., Timmis, J., Cooke, D., Neal, M., and King, C. (1999). *Artificial Immune Systems and Their Applications*, chapter Jisys: The Development on An Immune System for Real World Applications, pages 157–184. Springer-Verlag.

[Jain et al., 1999] Jain, A., Murty, M., and Flynn, P. (1999). Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323.

[Kanerva, 1988] Kanerva, P. (1988). *Sparse Distributed Memory*. MIT Press,Cambridge,MA.

[Kohonen, 1982] Kohonen, T. (1982). Self-organizing formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69.

[Potter and De Jong, 1998] Potter, M. and De Jong, K. (1998). The coevolution of antibodies for concept learning. In *Parallel Problem Solving From Nature - PPSN V*, pages 530–540. Springer-Verlag.

[Potter and De Jong, 2000] Potter, M. and De Jong, K. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29.

[Smith et al., 1999] Smith, D., Forrest, S., and Perelson, A. (1999). *Artificial Immune Systems and Their Applications*, chapter Immunological Memory is Associative, pages 105–112. Springer-Verlag.

[Timmis and Neal, 2001] Timmis, J. and Neal, M. (2001). A resource limited artificial immune system for data analysis. *Knowledge Based Systems*, 14(3-4):121–130.

[Timmis et al., 1999] Timmis, J., Neal, M., and Hunt, J. (1999). Data analysis using artificial immune systems, cluster analysis and kohonen networks: Some comparisons. In *Proceedings of the International Conference on Systems, Man and Cybernetics*, pages 922–927. IEEE.

[Timmis et al., 2000] Timmis, J., Neal, M., and Hunt, J. (2000). An artificial immune system for data analysis. *Biosystems*, 55(1/3):143–150.

Immune Memory in the Dynamic Clonal Selection Algorithm

J. Kim

Department of Computer Science
King's College London
Strand
London WC2R 2LS
jungwon@dcs.kcl.ac.uk

P. J. Bentley

Department of Computer Science
University College London
Gower Street
London WC1E 6BT
P.Bentley@cs.ucl.ac.uk

Abstract

The dynamic clonal selection algorithm (DynamiCS) was created to tackle the difficulties of anomaly detection in continuously changing environments (Kim and Bentley, 2002). This paper describes an extension to the original algorithm, involving the deletion of memory detectors that are no longer valid. Experiments are performed on the extended system and results are analysed. The results show a marked decrease in false positive errors produced by the system.

1 INTRODUCTION

A real computer network produces new network traffic continuously in real-time. Thus, normal behaviours of network traffic on one day can be different from normal behaviours of network traffic on another day. Previous work (Kim and Bentley, 2002), introduced the concept of an artificial immune system (AIS) based on a *dynamic clonal selection* algorithm (DynamiCS) to tackle this type of problem. This system is capable of learning normal behaviours by experiencing only a small subset of self antigens at one time. Its detectors were designed to be replaced whenever previously observed normal behaviours no longer represented current normal behaviours.

The results from experiments on this system (Kim and Bentley, 2002) showed that DynamiCS could incrementally learn the globally converged distributions even though only one subset distribution was given at each generation. This feature was achieved by employing three important parameters: *tolerisation period*, *activation threshold* and *life span*. However, DynamiCS could not learn new self-antigens when learned self and non-self behaviours suddenly altered due to legal self change. This resulted in high false positive (FP) rates when new antigens were monitored by DynamiCS, although it produced high true positive (TP) rates. The proposed explanation of this outcome was that the generated memory detectors had never been exposed to certain antigen clusters within their tolerisation periods. Thus they could not have tolerance against a complete self set.

This paper investigates a further extension of DynamiCS, so that it can reduce FP rates increased by memory detectors. As one way to decrease the FP rates caused by memory detectors, the extended DynamiCS handles generated memory detectors based on their detection results. DynamiCS preserved memory detectors for an infinite lifespan. In contrast, the extended DynamiCS presented here kills memory detectors if they show poor self-tolerance to new antigens. This extended system is tested to see whether surviving memory detectors no longer cause seriously high FP error rates or not. From this test, an analysis is performed to understand whether any other problems occur as a consequence of killing memory detectors. This paper is organised as follows: section 2 introduces the summary of DynamiCS algorithm and experimental results showing the role of memory detectors in DynamiCS. Section 3 reviews human immune memory and artificial immune memory. Section 4 presents the extended DynamiCS that adds the deletion of memory detectors to DyanmiCS. Section 5 finally concludes this paper.

2 DynamiCS REVISITED

2.1 ALGORITHM

The new AIS introduced in (Kim and Bentley, 2002) follows the basic concept of the AIS proposed by Hofmeyr (1999). The adaptability of Hofmeyr's AIS was achieved via co-ordinated dynamics of three different detector populations: immature, mature, and memory detector populations. In order to fully comprehend the co-ordinated dynamics of these three detector populations in terms of AIS adaptability, we introduced an artificial immune algorithm, called the *dynamic clonal selection algorithm* (DynamiCS). Although Hofmeyr proposed various new features in order to effect great adaptability and distributed detection, DynamiCS attempts to distill only the crucial components that yield adaptability to the system (and reduce the number of system parameters to ensure the algorithm is usable). This section presents the algorithm of the previous version of DynamiCS so that comparison can be made to the new version of this algorithm.

The following pseudo code provides an overview of the previous version of DynamiCS. DynamiCS starts by seeding initial immature detectors with random genotypes. DynamiCS then employs negative selection by comparing immature detectors to the given antigen set. As the result, immature detectors that bind to any antigens are deleted from the immature detector population and new immature detectors are generated until the number of immature detectors becomes the maximum size of the non-memory detector population. These same processes continue for the tolerisation period (*T*) number of generations. When the total number of generations reaches *T*, those immature detectors whose age reaches *T* (born at generation 1), become mature detectors.

```
Initialise Dynamic Clonal Selection Algorithm
Create an initial immature detector population with random detectors;

Generation_Number = 1;
Do
{
 If (Generation_Number = N) then Select a new antigen cluster.
Select 80% of self and non-self antigens from chosen antigen cluster;

Reset Parameters
    Generation_Number++;  Memory Detector Age++;
    Mature Detector Age++; Immature Detector Age++;

Monitor Antigens
{   Monitor Antigens by Memory Detectors
      Check whether any memory detector detects a non-self antigen;
      Check whether any memory detector detects a self antigen;

    Monitor Antigens by Mature Detectors
      Check whether any mature detector detects a non-self antigen;
      Check whether any mature detector detects a self antigen;
      Create new memory detectors;
      Old mature detectors are killed;

    Monitor Antigens by Immature Detectors
      Check whether any immature detector detects any self antigen;
      Delete any immature detector matching any self antigen;
      Create new mature detectors;
}

 If (immature detector population size +
    mature detector population size
    < non-memory detector pop size)
 {
  Do
    {  Generate a random detector;
       Add a random detector to an immature detector population;
    } Until  (immature detector population size +
              mature detector population size =
              non-memory detector pop size);
 }
} While (generation Number < max Generation)
```

At generation $T + 1$, a new antigen set is presented to the mature detectors to be monitored. Whenever a mature detector matches an antigen, the match count of a mature detector increases by one. After all the given antigens have been compared to all the existing mature detectors, the system checks: i) whether the match counts of mature detectors are larger than a pre-defined activation threshold (*A*) and ii) whether the ages of mature detectors meet a pre-defined life span (*L*). If there is a mature detector with a match count that is larger than *A*, this mature detector becomes a memory detector only if it indeed detects an intrusion. When a human security officer acknowledges that this detector detects any intrusion signature (costimulation), the detector activates and eventually becomes a memory detector. In addition, if the ages of mature detectors meet *L*, those mature detectors are deleted from the mature detector population.

At generation $T + 2$, when memory detectors match any antigen, confirmation is sought immediately from a human security officer. In this case, if the detected antigen patterns are confirmed as intrusion signatures, the detected antigen patterns are instantly deleted from the antigen set. After monitoring of new antigens by memory detectors, the remaining antigens are shown to mature detectors (if there are any). After the antigens have been monitored by the mature detectors, they are passed to immature detectors to perform negative selection. From generation $T + 3$ onwards, the same monitoring procedures that operated at generation $T + 2$ continue in order to monitor constantly changing antigen sets until the system terminates. For more detailed description about DynamiCS, readers are advised to refer to (Kim and Bentley, 2002)

All experiments used the Wisconsin breast cancer data set. The cancer data has two classes, 'Malignant' and 'Benign'. The system treated 'Malignant' as non-self and 'Benign' as self. In order to be sure of providing antigens of novel distributions, self and non-self antigen data was clustered into several groups using a clustering algorithm. The Expectation Maximization (EM) clustering algorithm was applied to cluster antigen data. The EM algorithm is widely-used as the basis for various unsupervised learning algorithms (Mitchell, 1997). We defined three as the number of generated clusters and this number was arbitrarily chosen. The EM algorithm divided the 240 'Malignant' examples into three clusters of 45, 117 and 78 examples, and the 460 'Benign' examples into three clusters of 42, 355 and 63 examples. 80% of the self and non-self antigen data belonging to each cluster were randomly selected for *N generations*. Here, 80% was arbitriarily selected. Therefore, DynamiCS was provided with different antigen data at each generation and the distributions of these data changed at every *N* generations. The costimulation mechanism involving a security officer was implemented by simply increasing the match count only when a detector detects non-self antigens.

2.2 DYNAMICS EXPERIMENTS

The experiments in previous work (Kim and Bentley 2002) simulated a situation in which converged behaviours learned in an incremental way are suddenly altered due to legal self change. The results of these experiments showed that a value of *T* which was sufficiently large to show perfect FP rates no longer demonstrated satisfactory FP rates. More precisely, four experiments were performed when four different values, {5, 10, 20, 30}, were given to *N*: the number of generations that antigens are selected from a same cluster.

In these four experiments, it was observed that the overall TP's and FP's increase as *N* grows. Particularly, when *N* is as large as a given *T*, which is tolerisation period, the obtained FP rates reached high values greater than 0.3. The inference from this result was that some memory detectors have never been exposed to a certain antigen cluster, and thus those memory detectors caused high FP rates. For this reason, the extension of DynamiCS introduced in this paper will handle generated memory detectors based on their detection results.

Before introducing the extended DynamiCS, another set of experiments was performed by giving different values for *A*, the activation threshold of a mature detector, but using smaller values than the ones used in (Kim and Bentley 2002). For consistency of experiments throughout this paper, another four experiments were performed with the parameter values shown in table 1.

Parameters	Values
Tolerisation Period (T)	30
Life Span of Mature Detectors (L)	10
Activation Threshold of Mature Detectors (A)	{5, 10, 20, 40}
Number of Generations that Antigens are Selected from the Same Cluster (N)	30

Table 1. Parameter values used for DynamiCS experiments

Figure 1 illustrates the results of these four experiments. The experiments were run five times and average results of five runs are shown in figure 1. The X-axes of these graphs represent the number of generations and the Y-axes indicate detection rates. Each graph has two lines, one displaying a True Positive (TP) rate and another showing a False Positive (FP) rate. The grid lines on the X axis were placed at every *N* generations for *N* = 30. Each experiment was run for a maximum of 2000 generations.

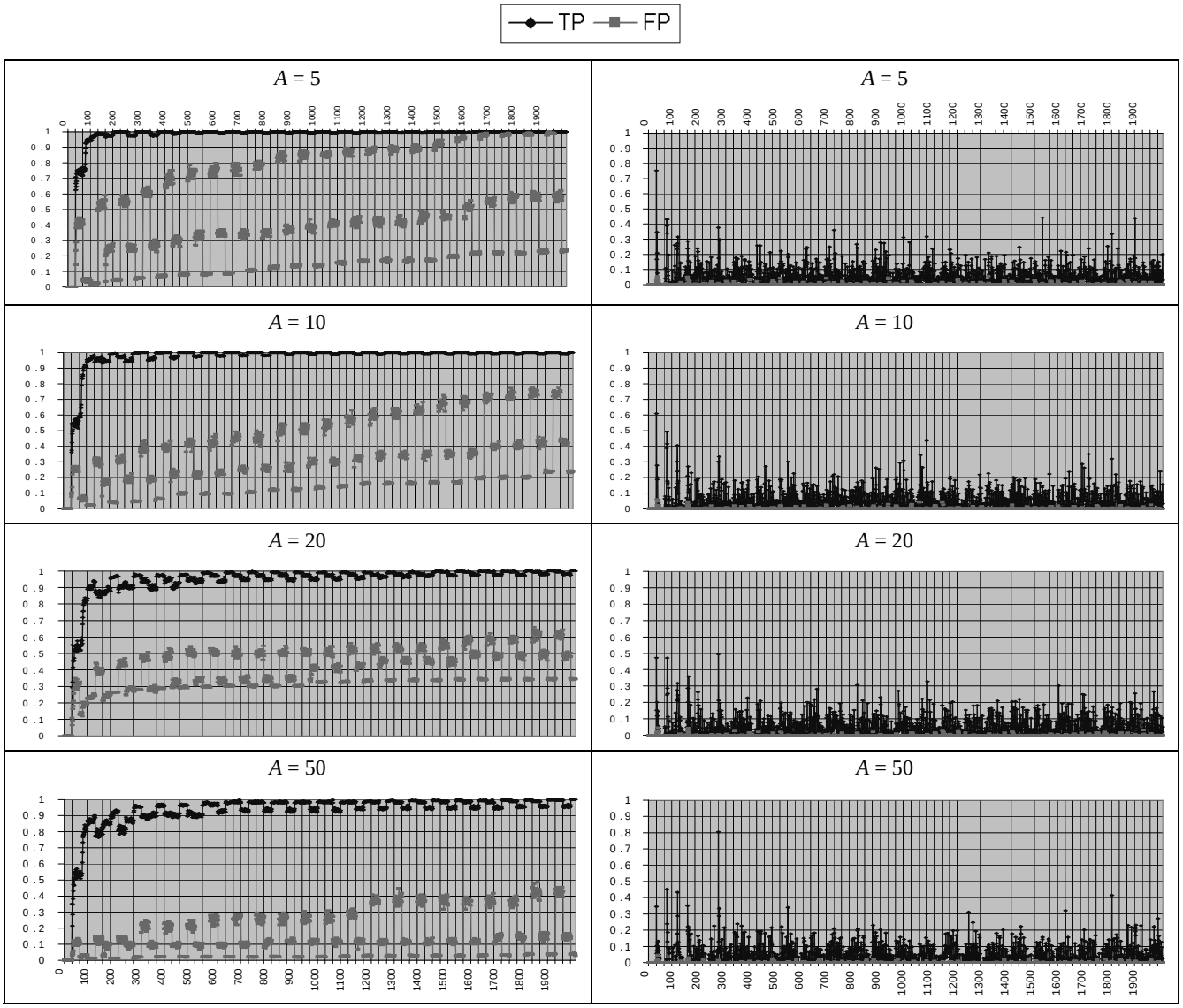


Figure 1. TP and FP rates when *A* varies and *T* = 30, *L* = 10, *N* =30

Figure 2. TP and FP rates when *A* varies and *T* = 30, *L* = 10, *N* =30 without memory detectors

It has been already seen that large *A* can prevent self antigen detection to some extent in (Kim and Bentley 2002). The results shown in figure 1 also follow the same consequence: large *A* reduces very high FP. Nevertheless, it still shows unacceptably high FP rates, around 0.5, even though FP rates dropped by nearly half when $A = 40$. This implies that the memory detectors which detected non-self antigens were not sufficiently self-tolerant.

However, there are two types of detector that are qualified to detect antigens: memory detectors and mature detectors that have just activated. Although it was inferred from previous experimental results that memory detectors which have never been exposed to a certain antigen cluster could cause high FP rates, it has not been shown yet whether memory detectors or mature detectors that have just activated are the actual cause of this problem. In order to clarify this issue, another set of experiments was performed. In the new experiments, DynamiCS did not generate memory detectors. When mature detectors activated, they produced detection signals but they were not converted into memory detectors. Instead, they simply died off. As the result, antigen detection was performed only by activated mature detectors in the absence of memory detectors. Figure 2 shows these new experimental results. The four important parameters, *T*, *A*, *L*, and *N*, have identical values to those used in the experiments above and they are summarised in table 1.

The four experimental results in figure 2 display similar outcomes regardless of *A* values: low TP and FP rates. This verifies the important role of memory detectors. They indeed contribute to increase TP rates by detecting re-encountering antigens. Without memory detectors, TP rates of DynamiCS fluctuate irregularly within an unsatisfying range (between 0.1 and 0.8). The low FP rates revealed in figure 2 also imply that the high FP rates shown in figure 1 are originated from detection by memory detectors.

The results exhibited in figure 2 makes it clearer that DynamiCS needs an appropriate way to handle memory detectors. In order to propose a new way of handling generated memory detectors, the next section briefly introduces how the human immune system maintains lifetime lasting memory while it continues to sustain self-tolerance. It also presents the method used to ensure that these human mechanisms have been implemented in available AIS's, in order to acquire artificial immune memory.

3 RELATED WORK: HANDLING MEMORY DETECTORS

3.1 HUMAN IMMUNE MEMORY

Immunologists define *immune memory* as the capability of the immune system that can fully protect the body from the re-attack of pathogens, which have previously been detected. Immune memory is long-lived, often lasting for many years, even for the lifetime of an individual (Tizard, 1995). The life–long immune memory means that a quick immune response to reappearing pathogens lasts for the lifetime despite constant and unpredictable generation of new memory cells, triggered by new antigen detection. Experimental observation showed that the memory cell population is maintained at a roughly constant size within an individual's body for his or her lifetime from puberty onwards (Yates and Callard, 2001), although there is gradual addition of new memory cells in old age. These two observations have raised a question of how the immune system maintains a roughly constant size of memory detectors while it continues to maintain immune memory of various types of pathogens that occur during a lifetime.

Several pieces of research by different immunologists attempted to explain the immune memory mechanism from various angles. For instance, one theory by Mackay (1993) explained this by showing the life-long lifespan of memory cells and another theory (Matzinger, 1994) described immune memory being provoked by constant re-stimulation of memory cells by reappearing antigens. In contrast to these theories, there is an observation of maintaining a roughly constant amount of memory cells in the absence of repeated exposure to antigens or cross-reactive stimulation (Yates and Callard, 2001). Yates and Callard showed that a small minority of memory cells are susceptible to programmed death triggered by contact with other memory cells. Especially when memory T-cells proliferate, matching the receptors of other T memory cells triggers signal cascade leading to programmed death of T memory cells. This theory showed that the regulation of a stable population of memory cells is achieved in absence of antigen stimulation.

These two interpretations can be combined together into one abstract explanation, which is the idiotype based immune network theory proposed by Jerne (1974). Immune network theory emphasises that the continuous chain of stimulation by antigens and suppression by other antibodies can form a large-scale network, and the final equilibrium status between suppression and stimulation determines the overall internal memory of the immune system. Therefore, the stabilised immune network constructed by proliferation by antigens and suppression by other antibodies constitutes a converged memory cell population.

Likewise, many studies in immunology approached the understanding of how to maintain a converged memory cell population as one step towards finding an explanation of lifetime lasting immune memory. Although there is no clear answer yet, the common explanation from these studies is that a memory cell population stabilises through constant death of existing memory cells, recruitment and proliferation of new memory cells. That is to say that a roughly constant size of memory cells is maintained not by keeping memory cells in a static way, but by continuous loss and new birth of memory cells in a dynamic way.

Although these studies illustrated how a stabilised memory cell population is maintained, they did not clearly explain how a stabilised memory cell population also maintains robust memory against various types of antigens and how it shows the associative property. The study by Smith *et al.* (1996) has attempted to explain the associative property of immune memory using Sparse Distributed Memory (SDM). SDM was originally introduced by Kanerva (1988) in order to store a very large number of data items into a memory space, which is mapped to a smaller number of physical data addresses. It approximately addresses given data items to a memory space when data is written. This means that the data item is recalled by an address sufficiently similar, but not necessarily equal, to the original address. This approximate addressing maps sparse and distributed physical addresses to logical addresses that is much more dense than existing physical addresses. Smith *et al.* (1996) took the view that distributed scattered physical addresses in SDM has an equivalence with memory cells in a stabilised memory cell pool, and that the approximated recalling mechanism of SDM is also equivalent to the primary and secondary responses of memory cells. Consequently, Smith *et al*. (1996) claimed that immune memory is robust and associative, as is SDM, since both employ a similar approximate addressing mechanism.

3.2 ARTIFICIAL IMMUNE MEMORY

The immune memory of the human immune system has been implemented in various ways in different AIS's. The common feature of these implementations is that the immune memory was achieved in an implicit way without having a separate memory detector/antibody population (Dasgupta, 1998; Timmis, 2001). Rather, only one type of antibody population was used and the antibody population was usually maintained at a constant size. That is to say that the antibody population was maintained through constant death of existing antibodies and recruitment and proliferation of new antibodies. During this process, naïve antibodies (i.e. newly generated) and a surviving antibody antibodies (i.e. memory antibodies) remain in the same antibody population and compete with each other for survival. Unlike the human immune system, where there are two different types of immune cell population (a memory cell and non-memory cell pool) and the competition between immune cells is only within each type of population (Yates and Callard, 2001), these AIS's did not label memory cells separately and thus they compete with other maturing and naïve immune cells for survival.

Among the AIS's which use only one antibody population, immune network theory has been a popular approach to make immune memory emerge by itself within an AIS (Timmis, 2001; Farmer *et al.,* 1986; Varela *et al*., 1988). The AIS's employing the network theory formed the immune network as the result of immune pattern recognition. The specific shape of the immune network described the immune memory of the given immune system. The memory that emerges, which is a stabilised network structure, was also used to handle a dynamic environment. When a new antibody is generated and inserted into already formed immune network, this new antibody competes with other ones that are already in the network. The new network formed by the surviving antibodies is expected to provide a new solution to a new environment without losing the solutions to the previous environment. This was possible because the AIS decides on surviving antibodies in the network not only by their antigen stimulation level but also by their antibody suppression level. Although some antibodies did not receive a sufficient degree of stimulation from new antigens, they would not be deleted as long as they were not the subject of large suppression from other antibodies. These antibodies can remain and act as memory cells in the AIS.

Another type of immune memory employed for AIS's is SDM (Smith *et al.*, 1996). Hart and Ross (2001) adopted SDM in their co-evolutionary GA to cluster moving data. Immune memory was not explicitly implemented as a separate antibody population in either work. Instead, the SDM was used for antibody and antigen matching and recall. It lets each antibody vote (i.e. match and recall) several antigens instead of one antigen. Thus, when a new antigen is presented, the democratic result from all antibodies decides the label of the antigen, whether self or non-self. This kind of antibody and antigen matching and recalling mechanism showed an implicit immune memory feature by allowing one antibody to match more than one antigen.

Another work by Gaspar and Collard (1999) investigated an artificial immune system for a time dependent optimisation (TDO) problem. Their simple artificial immune system (Sais) was implemented by adding several immune system features (such as clonal selection, immune network theory, hypermutation) to a conventional GA. In this work, Gaspar and Collard have shown what affected system robustness, obtained through immune memory. *Robustness* is the ability to maintain diverse optima without losing previously encountered optima. This feature was expected to allow the system to provide solutions quickly when previously presented optimal functions are later given as targets. The experimental results illustrated that Sais showed stronger robustness than other types of GA. The stronger robustness of Sais was achieved by memorising previously found optima using idiotoype immune network selection. However, the good improved robustness resulting from the memory of previously encountered optima, was not maintained as the number of different optimisation targets increased.

In contrast to above approaches, Hofmeyr's AIS (1999) adopted a separate memory detector population that was isolated from other detector/antibody populations. Memory detectors in Hofmyer's AIS also had two significant features: quicker response and infinite life span. This system is the only AIS to provide immune memory by directly mimicking the memory cells of the human immune system. Immune memory was no longer

maintained implicitly in this system. Instead, it had explicit antibodies to retain memory of previously detected antigens, and these antibodies were treated differently from other antibodies. Although the initial life spans of memory antibodies were set to be infinite, they could be deleted when the number of existing memory antibodies reached the pre-defined maximum number. If the number of memory detectors was more than this number, randomly selected memory detectors were killed until the number of current memory detectors, including the newly generated memory detectors, reached the maximum number of memory detectors.

This section has introduced three different types of artificial immune memories: those based on network theory, SDM and an explicit memory population. Among them, an explicit memory population seems to have an advantage over the two types of implicitly emerging immune memory. As reported in Gaspar and Collard's work (1999), the AIS without an explicit memory population failed to maintain its memory when the number of required antibodies grew in order to cover all the existing niches in a solution space. This was because these antibodies competed with newly generated antibodies that were more stimulated by current antigens. It might always be more likely for new antibodies to dominate antibodies memorising past antigens, since their antigen stimulation level is always higher than the memory antibody's antigen stimulation level. When the number of required antibodies is not large, they can still remain in the antibody population alongside the new antibodies. However, when the number of required antibodies grows, they cannot always remain in the antibody population and thus the AIS prefer currently stimulated antibodies to other memory antibodies. Thus, some memory antibodies will be lost. On the other hand, memory antibodies in an explicit memory population do not compete with new antibodies and thus memory antibodies would not be lost as the expense of space for new antibodies. To benefit from this advantage, DynamiCS uses an explicit memory population to maintain memory detectors.

Although this work has extensively studied how the human immune system maintains its immune memory and also how AIS's obtain their memory, it is not very clear how either of these systems stop self-detection of previously generated memory detectors. However, there is still one suggestion from this study that can be directly used: replacing memory detectors. As Yates and Callard (2001) have shown, memory detectors are constantly replaced while the population size is roughly constant. Following this understanding, the extended DynamiCS constantly replaces memory detectors. The next section describes one approach to replacement of memory detectors, via memory detector costimulation.

4 EXTENDED DYNAMICS: KILLING MEMORY DETECTORS

4.1 ALGORITHM DESCRIPTION

All the generated memory detectors in DynamiCS have infinite life span and an activation threshold of one. However, this is quite different from what really happens to memory cells in the human immune system. Although memory cells have a much lower activation threshold and a longer life span than other maturing cells, the memory cell population stabilises through constant death of existing memory cells, recruitment and proliferation of new memory cells. The infinite life span of memory detectors adopted by DynamiCS is not a biologically inspired idea. Thus, instead of giving an infinite life span to generated memory detectors, the extended DynamiCS kills memory detectors based on their current detection results. If antigens that are newly detected by memory detectors turn out to be self-antigens, these memory detectors are deleted from the memory detector population. This modification mimics the costimulation of memory detector detection. To be precise, whenever a memory detector in the memory detector population detects any antigen, it asks for confirmation about whether the detected antigen is self or non-self from a human officer. It sends a detection signal only if the human officer confirms that the detected antigen is non-self, otherwise it is deleted. Thus, the extended DynamiCS deletes harmful memory detectors by applying costimulation to memory detectors as it does to activating mature detectors.

4.2 EXTENDED DYNAMICS EXPERIMENTS

Four different experiments were performed to test whether the extended DynamiCS can reduce the high FP rates observed in the previous experiment. The extended DynamiCS was set with the same parameter values used in the experiments reported in 7.3 DynamiCS Revisited and they are summarised in table 1.

Figure 3 illustrates the results of four different experiments. These results can be compared to those in figure 1. Regardless of *A*, all of these four results show reasonably low FP rates, which are mostly lower than 0.1. This outcome is clearly different from the ones seen in figure 1, which has much higher FP rates and was much more sensitive to various *A* values. In figure 1, as *A* increases, the FP rates drop rapidly. In contrast, the changes of the FP rates in figure 3 are not clearly noticeable depending on various *A* values. In addition, the TP rate changes found in figure 1 and figure 3 are quite different. The TP rates shown in figure 3 decrease to a much greater extent compared to the TP rate changes observed in figure 1. In summary, as *A* increases, the amount of FP rate drop is much larger in the experimental results of original DynamiCS, while the degree of TP rate fall is much larger in the experimental results of the extended DynamiCS.

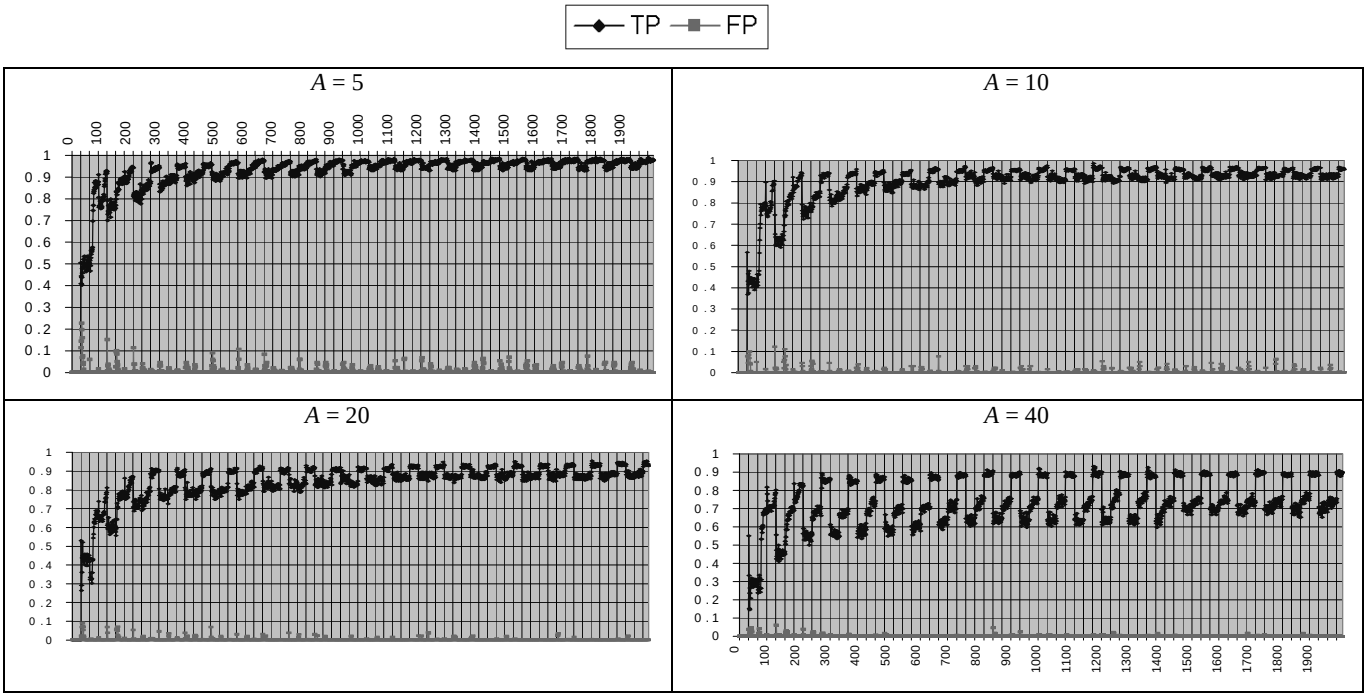


Figure 3. TP and FP rates when *A* varies and *T* = 30, *L* = 10, *N* =30 after killing memory detectors

Table 2. Average Number of Surviving, Generated and Deleted Memory Detectors per generation for DynamiCS and Extended DynamiCS. The values in parentheses are variances.

	DynamiCS			**Extended DynamiCS**			
	Surviving Memory Detectors	Generated Memory Detectors	Deleted Memory Detectors	Surviving Memory Detectors	Generated Memory Detectors	Deleted Memory Detectors	Memory Detecor CoStimulation per generation
A=5	75.75 (8.2)	75.75 (8.29)	0	46.5 (3)	205.5 (8.03)	159 (8.33)	40.89 (4.48)
A=10	49.5 (5.7)	49.5 (5.7)	0	32.75 (18.25)	124.25 (50.69)	91.5 (33.77)	29.36 (6.28)
A=20	33.5 (1)	33.5 (1)	0	24.25 (14.25)	78.75 (5.62)	54.5 (3.83)	20.39 (8.35)
A=40	20.5 (1.67)	20.5 (1.67)	0	14.5 (1.67)	55.25 (4.09)	40.75 (5.02)	16.43 (11.76)

These different effects explain how useful memory detectors in each system are for detecting new non-self antigens without mistakenly detecting self antigens. In the original DynaimCS, there are some memory detectors that detect self-antigens mistakenly and thus cause high FP rates. The generation of these memory detectors was prevented to some extent by restricting the conditions that allow mature detectors to be memory detectors. A large value for *A* in original DynamiCS did this job, and the large FP rate drop in figure 1 was obtained due to large *A*. Nevertheless, it has not yet gained satisfactory FP rates with a quite large value, 100, for *A* and also large *A* caused TP rates to decline. In contrast, for extended DynamicCS, memory detectors that caused high FP rates could not survive and thus FP rates were consistently low regardless of *A's* value. Extended DynamiCS only kept memory detectors that were useful for detecting non-self antigens without detecting self antigens. For the same reason, large *A* reduced the number of useful memory detectors and it resulted in lower TP rates.

Furthermore, the new strategy of the extended DynamiCS affects detection of non-self antigens. Compared to the original DynamiCS, it is much harder for memory detectors to survive in the extended DynamiCS. Table 2 shows the total number of surviving, generated and deleted memory detectors for a total of two thousand generations. These numbers are averaged across five runs. Thus, the average numbers of surviving memory detectors are smaller than the ones in DynamiCS when the same values are given to other parameters (see table 2). Consequently, the extended DynamiCS gains higher TP rates when it has a more relaxed condition for the activation of mature detectors, as in the cases having small values for *A* (see figure 3). Thus, the extended DynamiCS was able to obtain high TP rates and low FP rates when it had a small value for *A*.

However, there is another issue to be concerned with in the application of the extended DynamiCS for intrusion detection. Since the extended DynamiCS cured a problem of DynamiCS by applying costimulation to memory detectors, and costimulation was implemented in extended DynamiCS by asking for confirmation from a

human security officer, the large number of memory detector costimulations can hinder the adoption of the extended DynamiCS. Too much requirement for human intervention could render the extended DynamiCS useless. Thus, an effective IDS always requires the lowest frequency of costimulation per generation, leading to the least requirement for human intervention.

The amount of memory detector costimulation per generation governs the maximum number of activating detectors that will ask for detection confirmation from a security officer. Thus, it can be defined as the number of existing memory detectors per generation plus the number of mature detectors that have just activated (i.e. that just became memory detectors) per generation. The average numbers of memory detector costimulations per generation are shown in table 2. Although it is preferred for the extended DynamiCS to have smaller value of *A* because it leads to higher TP rates while sustaining low FP rates, this case tends to have larger number of memory detector costimulations. Thus, this approach, obtaining high TP rates and low FP rates by having small *A* values, does not seem to be ideal. Instead, these results suggest that large *A* can be more favourable than the case with small *A* if it can maintain a satisfactorily high TP rate. The experimental results require the extended DynamiCS to have a procedure to increase TP rates while it sustains a smaller number of memory detector costimulations. As one approach to this, future work investigates applying hypermutation for gene library evolution, as observed in the human immune system.

5 CONCLUSION

The experimental results in the previous work verified that DynamiCS could not learn new self-antigens when learned self and non-self behaviours are suddenly altered due to legal self change (Kim and Bentley, 2002). This resulted in high FP rates when new antigens were monitored by DynamiCS, although it produced high TP rates. The proposed explanation of this outcome was that the generated memory detectors had never been exposed to a certain antigen cluster within their tolerisation periods. Thus they could not have a sufficient degree of tolerance against a complete self set. For tackling this problem, this paper investigated a further extension of DynamiCS, so that it can reduce FP rates increased by memory detectors.

As one way to decrease the FP rates caused by memory detectors, DynamiCS was extended by eliminating memory detectors when they showed a poor degree of self-tolerance to new antigens. This extended system was tested to see whether surviving memory detectors no longer cause seriously high FP error rates or not. The test results showed that deletion of memory detectors based on their self-antigen detection dramatically decreased high FP rates that were observed in the previous paper. However, this method required a larger amount of costimulation in order to gain such benefits. The large amount of costimulation indeed can render the system weak for intrusion detection. This disadvantage demanded a further extension of DynamiCS.

In order to resolve this problem, further work studies the use of hypermutation to simulate gene library evolution. This additional extension is described in the sister paper to this paper, entitled: *A Model of Gene Library Evolution in the Dynamic Conal Selection Algorithm*.

References

Kim, J. and Bentley, P. J. (2002) "Towards an Artificial Immune System for Network Intrusion Detection: An Investigation of Dynamic Clonal Selection", *Proceedings of Congress on Evolutionary Computation*, pp.1015-1020, 2002.

Hofmeyr, S., (1999) *An Immunological Model of Distributed Detection and Its Application to Computer Security*, PhD Thesis, Dept of Computer Science, University of New Mexico, 1999.

Tizard, I. R., (1995) *Immunology:Introduction*, 4th Ed, Saunders College Publishing, 1995.

Yates, A. and Callard, R., (2001) "Cell Death and the Maintenance of Immunological Memory", *Discrete and Continuous Dynamical Systems* B **1**, pp.43-60, 2001.

Mitchell, T., *Machine Learning*, McGraw-Hill, 1997.

Mackay, C. R., (1993) "Immunological Memory", *Advanced Immunology*, vol. 53, pp.217-265, 1993.

Matzinger, P., (1994) "Immunological Memories Are Made Of This?", *Nature*, Vol. 369 No. 6382, pp.605-606, 1994.

Jerne, N. K., (1974) "Towards a Network Theory of the Immune System", *Annual Immunology (Inst. Pasteur)*, Vol.125, No.C., pp.373-389, 1974.

Smith, D. J., Forrest, S., and Perelson, A. S., (1996) "Immunological Memory is Associative", *Workshop Notes, Workshop 4: Immunity Based Systems, Int. Conference of Multiagent Systems*, Kyoto, Japan, pp.62-70, 1996.

Kanerva, P., (1988) *Sparse Distributed Memory*, MIT Press, Cambridge, M, 1988.

Dasgupta, D., (1998) "An Overview of Artificial Immune Systems and Their Applications", *Artificial Immune Systems and Their Applications*, (Ed) Dasgupta, D., Springer-Verlag, Berlin, 1998.

Timmis, J., (2001) *Artificial Immune Systems: a Novel Data Analysis Technique Inspired by the Immune Network Theory*, PhD Thesis, Dept. of Computer Science, University of Wales, Aberystwyth, 2001.

Farmer, J. D., Packard, N. H., and Perelson, A. S., (1986) "The Immune System, Adaptation and Machine Learning", *Physica* 22D, pp.182-204, 1986.

Hart, E., and Ross, P., (2001) "Clustering Moving Data with a Modified Immune Algorithm", *Proceeding of*

Applications of Evolutionary Computing, EvoWorkshops, pp. 394-404, 2001.

Gaspar, A., and Collard, P., (1999) "From Gas to Artificial Immune Systems: Improving Adaptation in Time Dependent Optimisation", *Proceeding of CEC99*, 1999.

Stable Clusters Formation in an Artificial Immune System

S.T. Wierzchoń

Department of Computer Science,
Białystok Technical University
ul. Wiejska 45[a], 15-351 Białystok, Poland
and
Institute of Computer Science,
Polish Academy of Sciences
ul. Ordona 21, 01-267 Warszawa, Poland

U. Kużelewska

Department of Computer Science,
Białystok Technical University
ul. Wiejska 45[a], 15-351 Białystok, Poland

Abstract

A new version of an artificial immune system designed for automated cluster formation in training data is presented. The algorithm fully exploits self-organizing properties of the vertebrate immune system and produces stable immune network. The algorithm uses the minimal number of control parameters.

1 INTRODUCTION

Artificial Immune System, or AIS, is a new, biologically inspired, paradigm of information processing. Its main principles are abstracted from the behaviour and properties of the vertebrate immune system, which is responsible for maintaining homeostasis of a living organism and particularly for protecting the organism from pathogens that could disrupt that homeostasis. More precisely, the immune system is a multi-layered structure. Each layer is of different complexity and the most complex is so-called *adaptive* immune system – consult (Hofmeyr, 2001) for details. In this paper by "immune system" we will understand the adaptive immune system. From a computer science perspective this last layer is a complex, self organizing and highly distributed system which has no centralized control and which uses learning and memory when solving particular tasks. The learning process does not require negative examples and the acquired knowledge is represented in explicit form.

The main actors of the adaptive immune system are B-lymphocytes (or B-cells) which mature in *bone marrow*, and T-lymphocytes (or T-cells) which mature in *thymus*. B-cells can be viewed as the commandos equipped with specialized weapon (i.e. antibodies attached to a single B-cell surface); each type of weapon is designed to fight different kind of enemy (i.e. pathogen or more precisely: antigen). B-cells can start their attack only after receiving signal from their commanders, i.e. subspecies of T-cells called helper T-cells, or Th-cells.

Thus, from a computer science point of view, Th-cells are responsible for Self/Non-self discrimination and the mechanisms governing Th-cells behaviour are used for designing novelty detection algorithms which can be used e.g. in the detection of computer viruses, or anomaly detection – consult (Dasgupta, 1999) for details. The "algorithms" used by B-cells (and reviewed in Section 2) are useful in adaptive data analysis, (Timmis, 2000), (De Castro and von Zuben, 2001), machine learning (Hunt and Cooke, 1996) or function optimisation (Bersini, 1990).

In this paper a new algorithm for adaptive clusters formation is given. Mentally based on the idea developed by Timmis (2000) the algorithm almost does not require control parameters and produces stable, long lived clusters. Section 3 describes this new algorithm and Section 4 contains numerical examples. Some general properties of the algorithm are discussed in Section 5.

A reader interested in models used in theoretical immunology is referred to the paper (Perelson and Weisbuch, 1997).

2 IMMUNE PRINCIPLES

Perhaps the first paper announcing exciting properties of the immune system was that of Farmer, Packard and Perelson (1986). As noted by Timmis (2000), the model described in this paper has shown how to use immune mechanisms in designing computer learning systems by: (i) using the idea of idiotypic network to achieve memory of what is being learnt, (ii) using a simple pattern matching mechanism between B-cell and antigen to define affinity, (iii) only representing B cells in the model and ignoring the effect of T cells, (iv) using a simple equation to model the stimulation of the B-cell, and (v) using mutation mechanisms to create diverse set of B cells. Let us briefly describe main mechanisms engaged during the immune response.

A single B-cell has about 10^5 receptors (antibodies) located on its surface. Each receptor has a specialized region, called *paratope*, used for identifying other molecules. Being a 3-D structure with uneven surface the paratope have a unique shape and other unique characteristics (e.g. van der Waals forces) referred to as the *specificity*. The regions on any molecule that the paratopes can attach to are called *epitopes*. If the two colliding molecules have complementary specificities,

they bind to each other and the strength of the bond (called *affinity*) depends on the degree of complementarity. A molecule bound by an antibody is referred to as the *antigen*[1]. A crucial role of the immune system is the binding of antibodies with antigens which serves to tag them for destruction by other cells. This process is termed *antigen recognition*. To treat formally the recognition problem, Perelson (1989) introduced the notion of the *shape space*. Namely, if there are m features influencing the interaction between the molecules (i.e. the spatial dimensions, charge distribution, etc.) and D_i is the domain of i-th feature ($i = 1, \ldots, m$) then each molecule is reduced to a point (the generalized shape of a molecule) in m-dimensional space ($S = D_1 \times \ldots \times D_m$. Typically S is a subset of m-dimensional Hamming space, or m-dimensional Euclidean space.

When a B-cell recognizes an antigen, it clones (i.e. produces identical copies of itself) as well as secretes free antibodies. The process of amplifying only those cells that produce a useful antibody type is called *clonal selection*, and the number of clones produced by a lymphocyte is proportional to its stimulation level. Clones are subjected to *somatic mutation* (characterized by high mutation rate) that results with new species of B cells having slightly different antibodies. These new B cells also bind to antigens and if they have a high affinity to the antigens they in turn will be activated and cloned. The rate of cloning a B-cell is proportional to its "fitness" to the problem: fittest cells replicate the most. The somatic mutation guarantees sufficient variation of the set of clones, while selection is provided by competition for pathogens. The whole process of (in fact: Darwinian) selection and differentiation of B-cell receptors leading to the evolution of B-cell populations better adapted to recognize specific epitopes is said to be *affinity maturation*.

Besides somatic mutation the immune system uses a number of other mechanism to maintain sufficient diversity and plasticity. Particularly about five percent of the B-cells are replaced every day by new lymphocytes generated in the bone marrow. This process is termed *apoptosis*.

The immune system possesses two types of response: primary and secondary. The *primary response* occurs when the B cells meet the antigen for the first time and reacts against it. To learn the structure of the antigen epitopes, clonal selection and somatic mutation are used. The primary response takes some time (usually about 3 weeks) to destroy the antigen. If the organism is reinfected with a previously encountered antigen, it will have an adapted subpopulation of B-cells to provide a very specific and rapid *secondary response*. From a computer science perspective the primary response corresponds to the identification of clusters in the training data, while the secondary response – to the pattern recognition problem, i.e. the assignment of a new data into one of existing clusters. Interestingly, the secondary response is not only triggered by the re-introduction of the same antigens, but also by infection with new antigens that are similar to previously seen antigens. That is why we say that the immune memory is *associative*. This phenomenon is modelled in the shape-space formalism by introducing so-called *recognition ball*, i.e. a ball B_r with radius r and centred in the point corresponding to the generalized shape of a given antibody.

The final immune system principle that plays a useful role in designing AIS's is that of *immune network* theory formulated by Jerne (1974), and further developed by Perelson (1986). According to this theory (called also *Jerne's hypothesis*) the immune response is based not only on the interaction of B-cells and antigens but also on the interactions of B-cells with other B-cells. These cells provide both a stimulation and suppression effect on one another and it is partially through this interaction that the memory is retained in the immune system.

The immune system is in permanent flux. The whole network is subjected structural perturbations through appearance and disappearance of some cell species. The introduction of new species is caused by somatic mutation, apoptosis, or combinatorial diversity (e.g. genetic operations used to produce new paratopes). A crucial issue is the fact that the network as such, and not the environment, exerts the greatest pressure in the selection of the new species to be integrated in the network. Thus, the immune network is self-organizing, since it determines the survival of newly created clones, and it determines its own size. This is referred to as the *meta-dynamics* of the system, (Varela and A. Coutinho, 1991).

The two most influential data analysis systems based on the immune metaphor are `aiNet` (De Castro and von Zuben, 2001) and `AINE` (Timmis, 2000). In both the systems the training set is identified with the set of antigens and the aim is to produce a set of B cells or antibodies representing these antigens.

According to Jerne's hypothesis, `AINE` produces networks (counterparts of idiotypic network) describing the key features of data items within the training set. The system uses almost all mechanisms described in this section, i.e. (i) it uses a set of B cells each of which is capable of recognizing antigens, (ii) similar B cells are linked together; these links form a network of B cells, (iii) clonal selection and hypermutation are performed on B cells, (iv) a number of B cells can be represented by an artificial recognition ball, or ARB. In fact, to improve stability of the immune network, AINE uses a population of ARBs and not the population of B cells. It needs four important control parameters: network affinity threshold (NAT), the mutation rate, the number of ARBs and the number of clones produced by a stimulated ARB. The influence of these parameters on final network is analysed in (Knight and Timmis, 2001).

The `aiNet` system, on the other hand, uses simplified representation: instead of B cells or ARBs it simply

[1] Antigen is a shorthand of *anti*bodies *gen*eration.

develops a population of antibodies. The population is initialised randomly (while AINE uses a random subset of antigens) and next it is modified by clonal selection, hypermutation and apoptosis. Interesting feature of the algorithm is that the clonal selection controls the network dynamics and metadynamics. Its main drawback is large number of user-defined control parameters. Further to obtain the immune network we have to use standard clustering tools: hierarchical clustering and graph-theoretic algorithms. But its advantage is very concise description of training data. In some cases such a data reduction equals 90% (Wierzchoń, 2001).

Both the algorithms are examples of *unsupervised* machine learning algorithms. Watkins (2001) used a combination of the just described approaches to design *supervised* learning algorithm. His aim was to develop a predictive model based on input data and the known classes in the data set.

To finish this section, let us note that the model of immune memory proposed by Jerne resembles the models of hypercycles or autocatalytic sets considered in the context of prebiotic chemical evolution – cf. (Bagley and Farmer, 1992) or (Eigen, 1971). It seems that careful examination of these models may be of value in constructing effective data analysis.

3 A NEW ALGORITHM

As stated in previous section, natural immune system contains B-cells with antibodies attached to their surfaces.

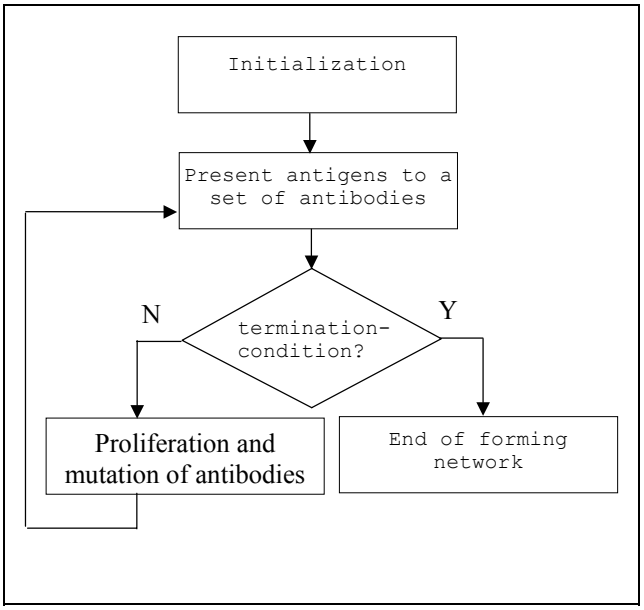


Figure 1. The algorithm for an immune network generation

In our AIN (artificial immune network) B cells are reduced to antibodies. This resembles the idea applied in the aiNet system. Let $\mathbf{Ab} = \{\mathbf{ab}_1, \ldots, \mathbf{ab}_n\}$ denotes the set of antibodies, and $\mathbf{Ag} = \{\mathbf{ag}_1, \ldots, \mathbf{ag}_k\}$ be the set of antigens. Each element $\mathbf{y} \in \{\mathbf{Ab}, \mathbf{Ag}\}$ is m-dimensional real-valued vector $\mathbf{y} = \{y_1, \ldots, y_m\}$.

The algorithm depicted on Figure 1 creates an AIN in the way similar to that used in (Timmis, 2000) but with some significant modifications. The nodes of the AIN represent antibodies, and their aim is representation of generalized characteristics of the antigens. Connected antibodies form clusters, so if any antibody from the cluster recognizes an antigen, it means the antigen belongs to this cluster. Recognition of an antigen, **ag**, by antibodies relies upon searching for an antibody **ab*** that minimizes Euclidean distance $d(\mathbf{ag}, \mathbf{ab}_i)$, $i = 1, \ldots, n$. The inverse of $d(\mathbf{ag}, \mathbf{ab}_i)$, can be viewed as the *affinity* of **ag** to $\mathbf{ab}_i$. Thus the smaller the distance $d(\mathbf{ag}, \mathbf{ab}^*)$ the better representation of **ag** by **ab*** is. The maximal length of an arc joining two nodes in the AIN is just the *NAT* scalar. Cells (antigens and antibodies) located further than *NAT* do not influence one another. The *NAT* parameter determines the granularity of the network and its overall connectivity (Knight and Timmis, 2001).

3.1 INITIALIZATION OF THE AIN

Like in (Timmis, 2000) this process is divided into three stages. First, the training data are normalized, i.e. the set **Ag** becomes a subset of m-dimensional unit cube $[0, 1]^m$. The second stage is to calculate an initial NAT value using the **Ag** set. Timmis calculates it as $\langle d \rangle \cdot \alpha$, where $\langle d \rangle$ is the average distance between each item in **Ag** and $\alpha \in (0,1\rangle$ is a constant. Such computed parameter was fixed during whole process of network creation. However such a procedure requires several runs of the algorithm and necessity of choosing the best value. In our approach the NAT value is computed in every iteration of the algorithm without external intervention. Initial NAT value is computed as follows. Let

$$D = \{d(\mathbf{ag}_i, \mathbf{ag}_j): i = 1,\ldots, k\text{-}1,\ j = 2, \ldots, k, j > i\}$$

be the set of distances values between each unique pair of antigens. Sort ascending the elements of D and denote

$$D' = \{d^1, \ldots d^l\} \quad (1)$$

a subset of D consisting of $l \le k \cdot (k-1)/2$ initial elements. Now the NAT is computed as the average value of the distances in the set D'. The third stage is to construct initial immune network. That is the set of antibodies **Ab** (also a subset of $[0, 1]^m$) is *randomly* initialised. This is in contrast with Timmis' approach. Next two antibodies are joined together only if their distance is not greater than the NAT value.

3.2 PRESENTATION OF THE ANTIGENS

In this stage every member of the **Ag** set is presented to the network and the stimulation level of each antibody, $sl(\mathbf{ab}_i)$, $i = 1,\ldots, n$, is computed.

According to Jerne's hypothesis the stimulation level of a B-cell is the sum of three factors: its affinity to the antigens, its affinity to its neighbours in the network and its enmity to these neighbours. This definition of the

stimulation level was applied in the AINE system. On the other hand, in the aiNet only the affinity of each antigen to all antibodies was taken into account. Similar idea was implemented in our system AIN. Define namely $\delta_i = \min_j d(\mathbf{ab}_i, \mathbf{ag}_j)$ to be the minimal distance between i-th antibody and the set of antigens. Now if $\delta_i \le$ NAT then $sl(\mathbf{ab}_i) = 1 - \delta_i$, and $sl(\mathbf{ab}_i) = 0$ otherwise.

Knowing the stimulation level of each antibody we can implement apoptosis, i.e. we can define the set of *effective* antibodies, $\mathbf{Ab}^* \subseteq \mathbf{Ab}$. Initially $\mathbf{Ab}^* = \mathbf{nil}$. First of all antibodies with zero stimulation value are removed from the set **Ab**. Denote **Ab**′ reduced set. Next, for each antigen, **ag**, we determine the set of antibodies recognizing this **ag**. If a given antigen is recognized by the unique antibody **ab*** then add this **ab*** to the set **Ab***. Hence **Ab**′ – **Ab*** is the set of potentially redundant antibodies. If $\mathbf{Ab}'' \subset (\mathbf{Ab}' - \mathbf{Ab}^*)$ is the set of antibodies each of which recognizes a group of identical antigens, we find a single antibody with highest stimulation value; only this antibody is moved to the set **Ab***.

This way we are still in the frames of an idiotypic network. Stimulation value awards antibodies with highest affinity to the antigens, while suppressive mechanisms are moved to the purging procedure. It seems that correctly designed purging procedure is responsible for generation of stable immune networks. Nasaroui, Gonzales and Dasgupta (2002) introduced fuzzy ARBs to improve stability of the immune networks. In our opinion it is not necessary. We can even use "standard" definition of stimulation level as proposed by Jerne and we can still generate stable networks provided that efficient antibodies elimination (described above) is implemented – see Sect. 4 for numerical results.

3.3 PROLIFERATION

Again this process is divided into three stages. First, the NAT value is recalculated using the cells from the set **Ab***. To do so, the set D' – see Eqn. (1) – is constructed; its cardinality is $l' \le l$.

Second, most stimulated cells from the set **Ab*** are cloned and mutated. In cloning process an antibody with stimulation level sl produces $\lfloor c_{max} \cdot sl \rfloor$ clones, where c_{max} is a constant (maximal number of clones). Clones are added to a separate set of clones **C**. Each clone $\mathbf{c} = (c_1, \ldots, c_m)$ is subjected mutation according to the equation

$$c_i = c_i + r \cdot \Delta,\ i = 1, \ldots, m$$

where r is a random number from the unit interval and $\Delta = 1 - y_i$ or $\Delta = -y_i$ (the decision which Δ value to choose is made randomly).

Third, mutated clones from the set **C** are integrated with the network, i.e. $\mathbf{Ab} = \mathbf{Ab}^* \cup \mathbf{C}$. Finally the immune network is reconstructed: two antbodies $\mathbf{ab}_i$, $\mathbf{ab}_j$ are joined together only if $d(\mathbf{ab}_i, \mathbf{ab}_j) \le$ NAT.

4 EXPERIMENTS

To verify the quality of this new algorithm, three data sets were analysed. In each experiment we focused on two-dimensional data representing two separate clusters. The first experiment is concerned with linearly separable clusters (Figures 2a-2d). In two remaining experiments, Figures 3a-3d and 4a-4d, training data exhibiting non-trivial patterns were used. Every AIN was developed through 50 iterations to observe stabilization of the NAT value as well as stabilization of resulting network size.

Every set of figures, denoted a – d, includes: (a) antigen set, (b) final network structure, (c) evolution of NAT and (d) evolution of the network size. In every case after some number of iterations network becomes stable. This number depends on size and complexity of antigen set.

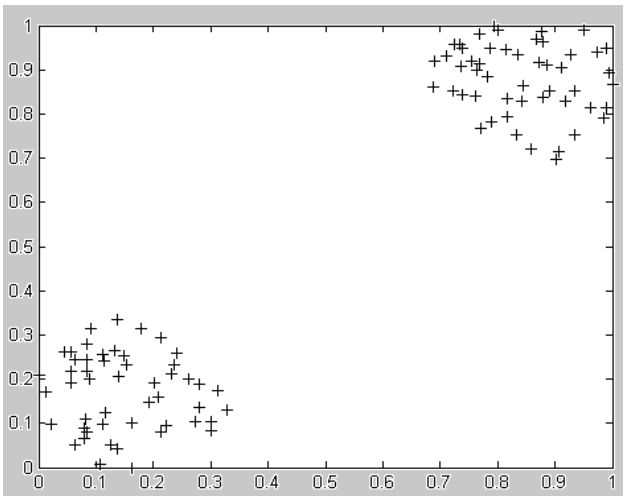


Figure 2a: Antigens set

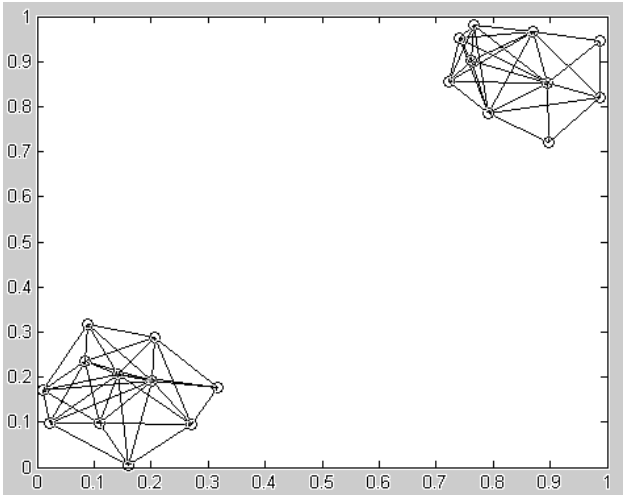


Figure 2b: Final immune network

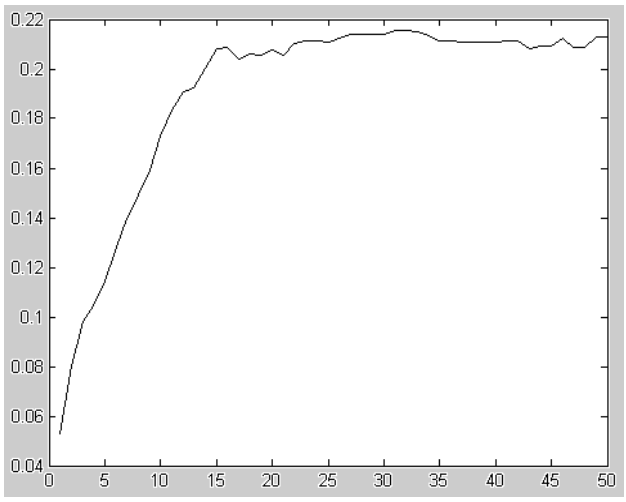


Figure 2c: Evolution of the NAT value

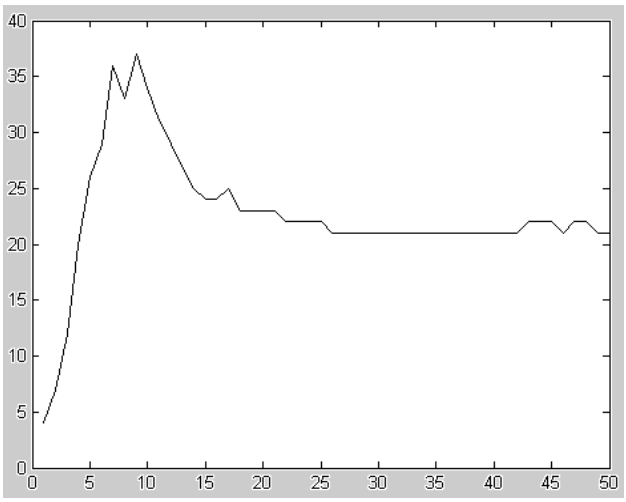


Figure 2d: Evolution of the network size

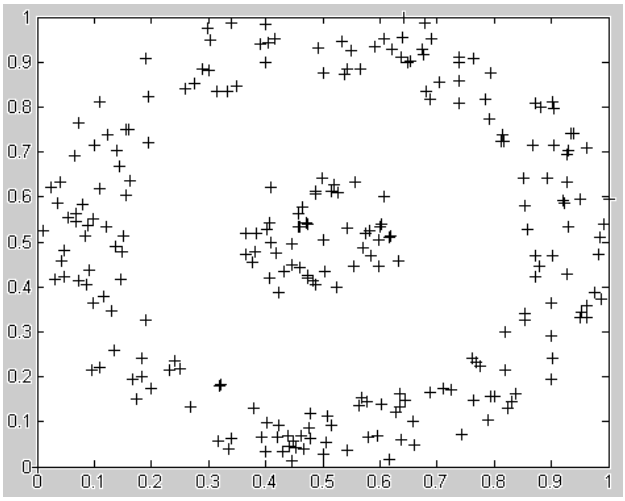


Figure 3a: Antigen set

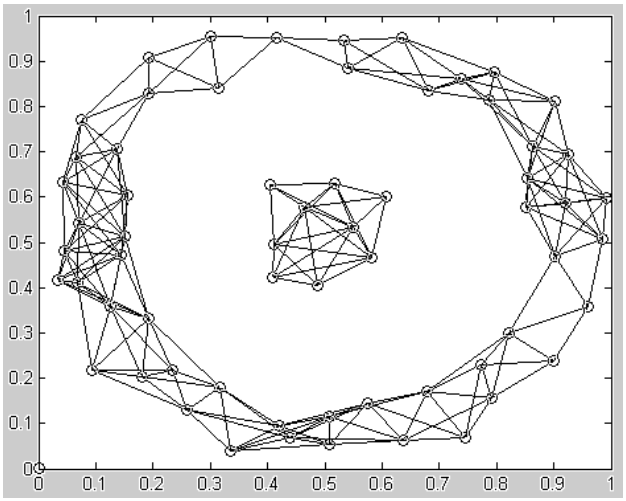


Figure 3b: Final immune network

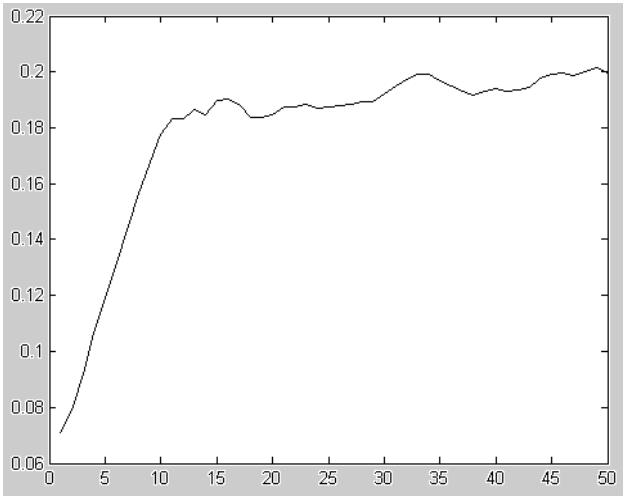


Figure 3c: Evolution of the NAT value

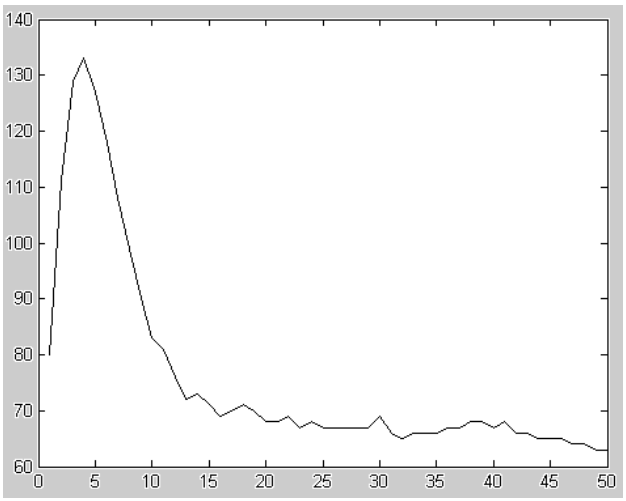


Figure 3d: Evolution of the network size

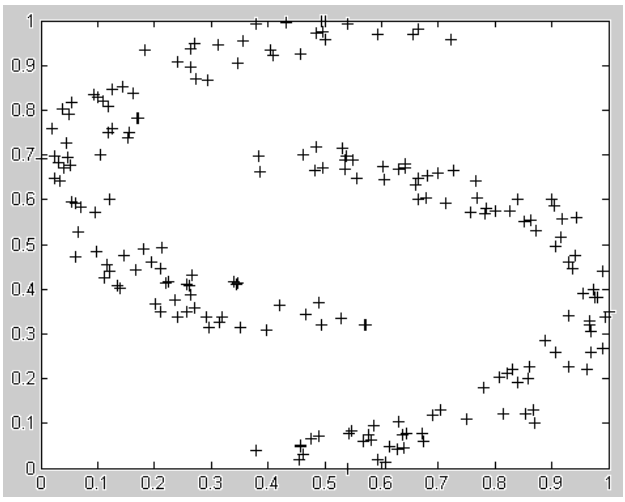


Figure 4a: Antigen set

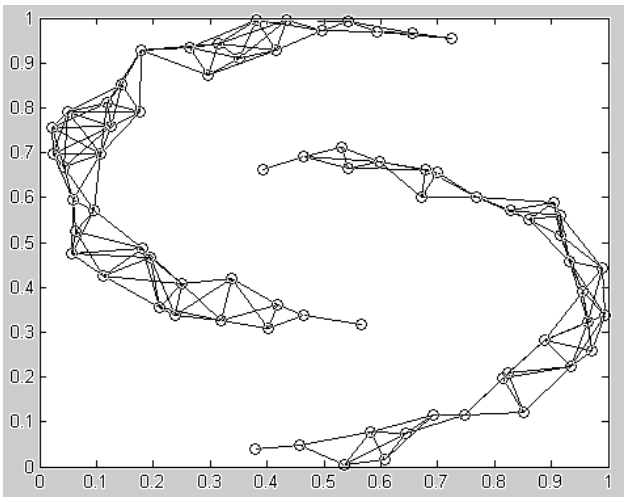


Figure 4b: Final immune network

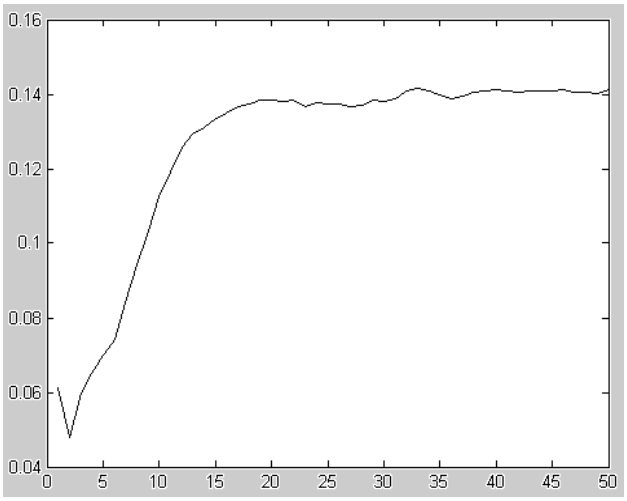


Figure 4c: Evolution of the NAT value

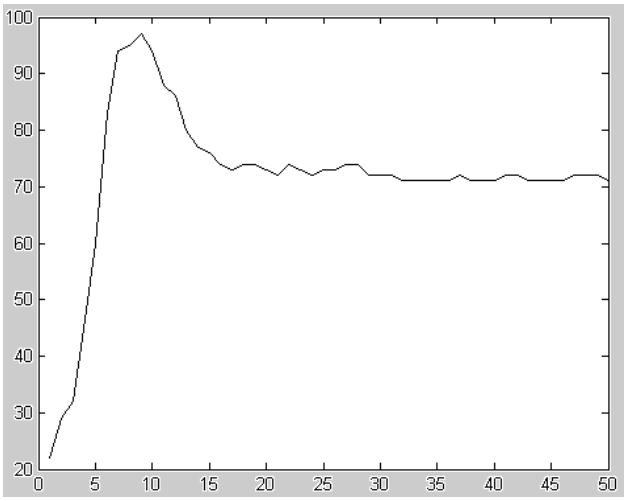


Figure 4d: Evolution of the network size

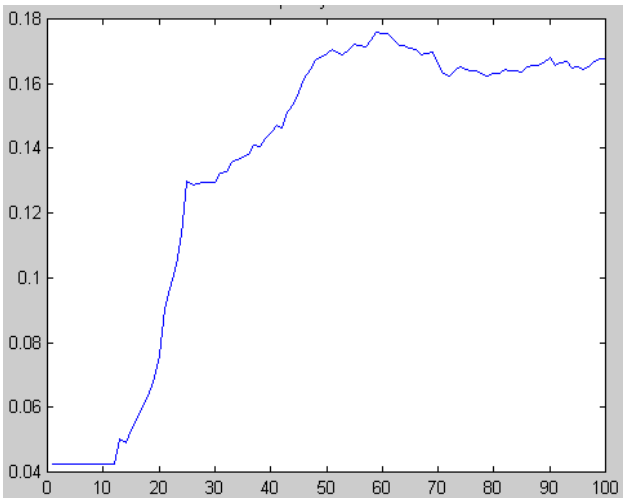


Figure 5a: Evolution of the NAT value in a system with "standard" definition of stimulation level

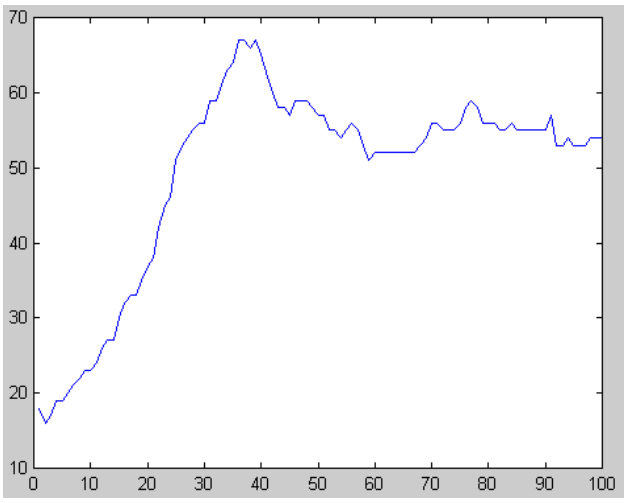


Figure 5b: Evolution of the network size in a system with "standard" definition of stimulation level

Interestingly, the algorithm behaves almost identical when the stimulation level is defined as in (Timmis, 2000). Figures 5a and 5b demonstrate evolution of the NAT value and network size for the antigen set presented

on figure 4a. In fact minor modifications of the purging procedure results in different behaviour of the algorithm (see the url: http://www.ipipan.waw.pl/~stw/ais for different data sets analysed with different purging strategy).

5 CONCLUSIONS

In all the cases analysed the algorithm is able to produce correct networks after 15-20 iterations. After this time the network structure becomes stable – its size and the NAT value oscillates around fixed value. This fact can be used as the definition of the termination condition. Final network structure represents *immune memory* – it can react faster and better when similar data are encountered in the future.

Interestingly, in each case we can observe data compression phenomenon like in the `aiNet` system. In first experiment the antigen set consists of 100 items and final immune network consists of 21 cells; so data compression ratio is 79%. In the second case data compression attains 69%, and in the third case – 29%. This property results from the second stage of the purging procedure (i.e. further reduction of the set **Ab′** described in Section 3.2). The compression ratio depends on the topology of input data.

The most important feature of the immune system is its ability to recognize new pathogens. The algorithm described in this paper passes this examination very well. The number, shape and location of generated clusters precisely reflects topological properties of the training set.

Additionally clusters are formed adaptively. This is in contrast to the aiNet system, where antibodies are generated first, and next graph theoretical methods are used to cluster these antibodies.

Finally the algorithm requires minimal number of control parameters indeed: it is necessary to define only the cardinality of the set D' – cf. Eqn. (1) – and maximal number of clones c_{max}. The NAT value evolves during subsequent iterations of the algorithm. The same applies to the network size.

6 FUTURE WORK

The algorithm described in this paper possesses many intriguing properties. Detailed mathematical analysis is necessary to confirm these properties. Particularly deeper analysis of the influence of stimulation level computation and purging implementation on the algorithm behaviour should be performed.

It was also observed (results not reported here) that in case of overlapping clusters the algorithm displays a kind of cross-reactive memory; this phenomenon should also be carefully verified.

Lastly, the algorithm behaviour on more complex data sets will be examined, and more flexible strategies for choosing effective antibodies **Ab*** will be worked out.

Acknowledgments

First author would like to thank Leandro Nunes de Castro and Jonathan Timmis for exhaustive comments on their systems as well as to the unknown referees for constructive remarks.

References

R. J. Bagley, and J. D. Farmer (1992). Spontaneous emergence of a metabolism. In: C.G. Langton, C. Taylor, J.D. Farmer, S. Rasmussen, eds., *Proc. of the Workshop on Artificial Life* (*ALIFE'90*), vol. 5 of *Santa Fe Institute Studies in the Sciences of Complexity*, Addison-Weseley 1992, 93-140

H. Bersini (1990) Hints for adaptive problem solving gleaned from immune networks. In: *Proc. of the First Workshop on Parallel Problem Solving from Nature*, LNCS 496, 343-354. Berlin: Springer-Verlag.

D. Dasgupta, ed. (1999) *Artificial Immune Systems and Their Applications*. Berlin: Springer-Verlag.

L. N. De Castro, and F. J. von Zuben (2001) aiNet: An artificial immune network for data analysis. In: H.A. Abbas, R.A. Sarker, Ch.S. Newton, eds., *Data Mining: A Heuristic Approach*, 231-259. Idea Group Publishing, USA

M. Eigen (1971) Selforganization of matter and the evolution of biological macromolecules. *Naturwis-senschaften* **58** : 465-523

J. D. Farmer, N. H. Packard, and A. S. Perelson (1986) The immune system, adaptation, and machine learning. *Physica D* **22**:187-204

S. A. Hofmeyr (2001) An interpretative introduction to the immune system. In L.A. Sagel and I.R. Cohen (eds.) *Design Principles for the Immune System and Other Distributed Autonomous Systems*, 3-26. New York: Oxford University Press.

J. E. Hunt, and D. E. Cooke (1996) Learning using an artificial immune system. *J. of network and Computer Applications*, **19**: 189-212

N. K. Jerne (1974) Towards a network theory of the immune system. *Ann. Immunol* (*Inst. Pasteur*) **125C**: 373-389

T. Knight, and J. Timmis (2001) Assessing the performance of the resource limited artificial immune system AINE. Technical Report No. 3-01. Computing Laboratory, University of Kent, Canterbury, Kent, UK.

O. Nasaroui, F. Gonzales and D. Dasgupta (2002) The fuzzy artificial immune system: Motivations, basic concepts, and application to clustering and Web profiling. In *Proc. of the IEEE International Conf. On Fuzzy Systems at WCCI*, pp. 711-716, Hawaii, May 12-17

A. S. Perelson (1989) Immune network theory. *Immunological Reviews*, **110**: 5-33

A. S. Perelson, and G. Weisbuch (1997) Immunology for physicists. *Reviews of Modern Physics*, **69**: 1219-1265

J. I. Timmis (2000) Artificial immune systems: A novel data analysis technique inspired by the immune network theory. Ph. D. Thesis. Department of Computer Science, University of Wales, Aberystwyth, September 2000

F.J. Varela, and A. Coutinho (1991) Second generation immune networks. *Immunology Today*, **12**: 159-166

A. B. Watkins (2001) AIRS: A resource limited artificial immune classifier. M. Sc. Thesis. Department of Computer Science, Mississippi State University, December 2001

S. T. Wierzchoń (2001) *Artificial Immune Systems. Theory and Applications* (in Polish). Akademicka Oficyna Wydawnicza EXIT, Warszawa 2001. ISBN 83-87674-30-3, 282+vii pp.

An artificial immune system for continuous analysis of time-varying data

Dr. Mark Neal,
Department of Computer Science,
University of Wales,
Aberystwyth,
Ceredigion, SY23 3AF
U.K.
Email: mjn@aber.ac.uk

Abstract

This paper presents an artificial immune system (AIS) which produces artificial immune networks that are meaningful, of a bounded size and dynamic over a very large number of data presentations. This behaviour had proved elusive up to this time but has now permitted the application of the AIS to situations requiring continuous learning. It also removes the need to decide when to stop training an AIS. The new version of the algorithm is described, and results are presented for analysis of static and dynamic versions of a trivial two-dimensional data set and Fisher's Iris data. It is argued that the changes made from previous versions of the "resource limited" algorithm are in keeping with the goals of remaining true to the immune system analogy and making the system as simple as possible.

1 INTRODUCTION

The human immune system is a complex natural defence mechanism that recognizes and responds to the presence of foreign substances (pathogens). The response elicited depends on the previous experience of the immune system in question. Invaders that display antigens (features of pathogens) that have been experienced previously elicit a more rapid and more powerful response. This flexibility enables the immune system to remove a huge variety of infections, many of them novel to the immune system in question. This ability to learn and respond to a wide variety of similar but different pathogens has roused the interest of Artificial Intelligence researchers who wish to learn from, emulate and exploit *artificial* immune systems.

There are several competing theories as to how the human immune system achieves the adaptability and flexibility that allows it to function so effectively. The existence and participation of the bone marrow, B-cells and T-cells in the process is beyond dispute. The ways in which these entities reproduce, clone and mutate is still a fertile field of study for immunologists. Computer scientists have for many years used evolutionary computing as a stock in trade (see Goldberg 1989), and thus understand something of how to deal with simulations of simple versions of these types of activity. The added interest of the immune system is in the mechanism that makes it so effective and so rapid in adapting, more rapid than organism level evolutionary adaptation.

Of the various mechanisms suggested, the network theory (see Jerne 1974, Perelson 1989), still very contentious in immunology circles, stands out as a tractable and familiar way to try to improve upon the performance of the standard genetic algorithm. AI has often resorted to networks of one type or another as mechanisms that can be made to exhibit emergent behaviour in a reliable, comprehensible and visually presentable way. Thus we have been working with models of immune systems based on network structures with B-cells as the primary unit (see Timmis et al. 1999, Timmis et al. 2000 and Timmis et al. 2001).

2 REAL AND ARTIFICIAL IMMUNE SYSTEMS

At this point a brief summary of some of the relevant terms and how they apply to real and artificial immune systems is appropriate:

i) Pathogen: for the biological immune system a pathogen is usually a foreign body such as a virus, bactaerium, fungus or other parasite. For an artificial immune system a complete data item represents a pathogen.

ii) Antigen: a real antigen is a substance which elicits a response from lymphocytes. These are often toxins or proteins which are characteristic of particular types of pathogen. In the artificial immune system a field within a data item with a particular value is comparable; as it is particular values in particular fields which stimulate the nodes in an artificial immune system.

iii) Lymphocytes: are the white blood cells in the real immune system which are responsible for the destruction of pathogens. B-cells and T-cells are two types of lymphocyte. In our artificial immune system B-cells are not represented individually,

but gathered together using the concept of the artificial recognition ball (ARB) as is described below (see section 2.4).

iv) Innate versus adaptive immunity: innate immunity does not change throughout the lifetime of the individual and relies on different mechanisms from adaptive immunity which is what we are concerned with and wish to emulate in our artificial immune systems.

2.1 INITIAL INNOCULATION

The adaptive human immune system is primed at a very early stage in various ways including from the mother's milk and via vaccinations. For the human these very early additions to the immunological repertoire often mean the difference between life and death. Clearly the ability to bootstrap the immune system before any dangerous pathogens are missed is an essential feature of any immune system. Fortunately the effects of failure in AI systems tend to be less drastic than in the human body, but nonetheless the sensitivity of any immune system, real or artificial, to its initial pre-programmed repertoire is of the utmost importance. If it is necessary to pre-program with a very large number of antigens, and the system is not capable of dealing with antigens significantly different from those in the initial innoculation then this is not satisfactory. In fact the less that is necessary to begin with, the better.

2.2 PRIMARY RESPONSE

The primary response of an immune system is provoked when an antigen not previously encountered is detected. The bone marrow will generate a large number of B-cells, in the expectation that some of them will be able to deal with the infection, and will thus take over the production of more and more effective antibodies. After the response has cleared the infection, some of the more effective B-cells produced will remain in the body ready to respond the next time a similar infection occurs.

This part of the process is recognized as a learning phase in which previously unseen patterns are stored for later recall. The way in which the B-cells that remain in the system are maintained, and do not die off is of fundamental importance and is where the network theory provides one of several possible answers.

2.3 SECONDARY RESPONSE

The secondary response is the response elicited when a familiar antigen is detected. Those B-cells already present in the body which are well adapted to dealing with the antigen will reproduce very rapidly to deal with the infection.

The secondary response can be seen as the recall phase in the artificial immune networks presented.

2.4 THE IMMUNE NETWORK THEORY

The immune network theory proposes that the B-cells in the body interact with each other to maintain the immune memory. The mechanism proposed is that B-cells which are capable of recognising similar (but not necessarily identical) pathogens are also capable of recognising and stimulating each other (see Farmer et al. 1986). Thus a dynamic feedback mechanism can maintain parts of the immunological memory which are not frequently stimulated. Clearly however not all B-cells have sufficient stimulation to survive indefinitely and thus some will die out.

In the human immune system T-cells both perform a surveillance role and interact with B-cells which complicates the mechanism somewhat. In our artificial immune system the role of T-cells is currently ignored.

In the real immune system there are very large numbers of identical B-cells to deal with each type of infection. In an artificial system such repetition can be coded without representing all the identical cells individually. Fortunately the concept of a *recognition ball* which represents a region of antigen space that is covered by a particular type of B-cell can replace the repetition of individuals (Perelson 1989).

So our AIS consists of a network of *artificial recognition balls* which are linked together if they are close to each other in antigen space. Pathogens (data items) can be considered to be points in this antigen space, and thus proximity can be defined as a simple distance function. When a data item is presented to the network the node which is the most stimulated produces clones of itself, some of which are mutated to increase the diversity of the network's recognition capabilities. The stimulation level of each node is calculated based upon its reaction both to the data items and to those nodes to which it is connected (see section 4.1). Thus nodes which are severely mutated into remote regions of the antigen space (and thus sparsely or totally disconnected) will not survive unless they match data items which are not presently covered by the network in which case they will expand its repertoire.

3 BACKGROUND

In a previous publication (see Timmis et al. 2001) we presented a resource-limited version of the AIS as a step toward a continuous learning version of the AIS presented in (see Timmis et al. 1999, Timmis et al. 2000). This previous work was motivated by the need for an AIS that did not rely on the arbitrary selection of the number of times that a data set should be presented to it, and the realisation that any AIS that did require such control was not a good model of a biological immune system. There were however several problems with the solution that we proposed:

i) the mechanisms which governed the resource allocation were centralised in a very artificial way,

which was contrary to the distributed nature of the original AIS

ii) there was no "inertia" effect bound to the resources. Thus an ARB could gain or lose all of its resources in one pass through the network, which is quite unlike the biological immune system which takes time to build up immunity and time to lose it again.

iii) The nature of the calculations performing the resource allocation required the normalisation of the stimulation levels, which lead to some inelegant, lengthy and unnecessarily complex calculations after every iteration

iv) After several passes through the data set in question the network would begin to degenerate and fail to represent some of the data items

v) The algorithm did not lend itself to a genuinely continuous mode of operation as resource allocation was performed after each pass through the data set. This required an epoch-based (synchronous update) approach which creates a variety of problems if the network is to be used in a continuous mode.

After several attempts to modify the resource allocation mechanism it became clear that these problems were quite severe and were leading to a complex and arbitrary set of solutions. Thus a different approach was taken based on a simpler mechanism used after every data item presented.

4 THE SSAIS

This new approach lead to the self-stabilising artificial immune system (SSAIS) presented here. Artificial recognition balls (ARBs) are still used as the basic component of the network, and they are still linked together in the same way. The network affinity threshold is also calculated in the same way and serves the same purpose as in the original systems. The SSAIS differs from the resource limited artificial immune system (RLAIS) in several ways. The most important difference is that there is no fixed quantity of resources to be distributed centrally between the ARBs. The concept of resources is still present, but in an altered form. In the RLAIS the resources were allocated to ARBs by order of and in proportion to stimulation level. In the SSAIS resources are dealt with locally by each ARB. An ARB increases its own resource allocation each time it registers the highest stimulation for an incoming data item. The ARB increments its resource holding by adding its current stimulation level. Additionally, each time a data item is presented the resource level of every ARB decays geometrically. The balance between the decay of the resource level and the occasional boost received when an ARB "wins" is quite robust, and results in more densely populated areas of the data space supporting larger numbers of ARBs and more sparsely populated regions fewer ARBs. This results in emergent behaviour that is very similar to that of the original AIS and the RLAIS, but without the "one shot" constraint of the former and the normalisation, synchronous update and sorting requirements of the latter.

4.1 THE STIMULATION FUNCTION

In order to bound the growth of the resource level in any ARB (and thus in the network as a whole) it was necessary to bound the stimulation level. The simplest way to achieve this is to make a small modification to the ARB stimulation function. The stimulation function in previous systems (see Timmis et al. 2000) was made up of three components:

i) An excitation factor, *ps* based linearly on the Euclidean distance to the current data item (*p*):

$$ps = 1 - dis(p)$$

ii) An excitation factor, *ns* based on the distance to the neighbours around the ARB:

$$ns = \sum_{x=0}^{n} 1 - dis(x)$$

iii) A suppression factor, *nn* based on the distance to the neighbours around the ARB:

$$nn = -\sum_{x=0}^{n} dis(x)$$

In all equations the function *dis(a)* returns the Euclidean distance between the current node and the item a; and *n* represents the number of neighbours at the current node.

These components are simply summed. The second and third components are based on the neighbours of the ARB, and there is no limit to the number of neighbours an ARB can have. This poses a problem in the form of the potential for unbounded growth. Two variants on this stimulation function were experimented with. The first of which is the most obvious and is simply the same as above, but with parts ii) and iii) divided by the number of neighbours. This succeeded in bounding the growth of the resource levels in the network, but resulted in networks which had one extremely dense and active region and other totally static sections which were much less dense remainders of the original network created from the initialisation data. In order to examine this behaviour a second simpler function was used with surprisingly effective results. The neighbour suppression factor was discarded completely and only the excitation retained. This resulted in a simpler stimulation function made up of only two parts which are summed:

i) An excitation factor, *ps* based linearly on the distance to the current data item:

$$ps = 1 - dis(p)$$

ii) A normalised excitation, *ns* factor based on the distance to the neighbours around the ARB:

$$ns = 1/n \times \sum_{x=0}^{n} 1 - dis(x)$$

When used within the scheme presented here, this function yielded networks which attain a "dynamic stability" with all parts of the network producing some clones (see below), and varying their topology a little at a time, whilst retaining the overall structure and distribution throughout the data space.

4.2 ALLOCATING RESOURCES

In this version of the immune network algorithm, *resources* are simply recorded as a numerical value associated with each node. This number is used both to decide when to remove a node from the network (when the resources fall below a minimum threshold) and to decide how many clones to produce (more resources implies more clones). Whilst there is no longer a central notion of resource availability, it is still appropriate to think of the ARBs being limited by available resources. In this system the ARBs allocate their own resources only when justified by reacting the most strongly to a data item. The level of resources at an ARB that is not the most stimulated by data item (i+1) is geometrically decaying with each data presentation, thus:

$$\mathrm{R}(a)_{(i+1)} = dr \times \mathrm{R}(a)_{(i)}$$

where $\mathrm{R}(a)_{(i)}$ represents the level of resources present at ARB a after the presentation of i data items and dr represents the rate at which the resource level at an ARB decays. The level of resources at the ARB which is the most stimulated by data item (i+1) will be:

$$\mathrm{R}(a)_{(i+1)} = dr \times (\mathrm{R}(a)_{(i)} + \mathrm{SL}(a)_{(i+1)})$$

where $\mathrm{SL}(a)_{(i+1)}$ represents the stimulation level of ARB a (as defined in section 4.1) after the presentation of data item (i+1). Thus when an ARB is the most stimulated for an incoming data item it gives itself a boost in its resource level. These two conflicting effects balance to ensure the survival of ARBs that regularly have the highest stimulation level and the gradual demise of those that do not. The decay rate scalar dr provides an easy control over the size of the networks produced. The values used for dr in this work were 0.999 for the trivial data set and 0.9995 for the Iris data. Some initial experimentation with these values was undertaken which seemed to indicate that the value of dr is a sensitive control for the size of the population.

4.3 POPULATION CULLING

After each data item is presented to the network any ARBs that have resources less than a fixed threshold value (the *mortality* threshold) are removed from the population. The threshold value used in this work was 0.6 for all networks regardless of the data set in use. This was an arbitrary choice, and further work is required to ascertain the sensitivity and range of values for this parameter. The networks produced do not seem to be particularly sensitive to the threshold at which nodes are culled. The values for *mortaility* and the multiplier for the resource level for new clones are also arbitrary and require further investigation.

4.4 CLONING MECHANISM

The cloning mechanism for the SSAIS is slightly different from previous systems. When an ARB is the most active it is allowed to undergo cloning. The ARB produces clones at a rate which is proportional to the resource level at the ARB. The number of nodes produced is calculated as follows:

$$nc = \mathrm{R}(a)(i)/(mortality \times 10)$$

where *mortality* is the minimum resource level that a node can have before being culled. This is because each clone that is produced is assigned *mortality* × 10 resources from the ARB's pool of resources. As each clone is produced its data fields are mutated with a fixed probability (the mutation rate). The mutation rate was fixed throguhout this work at 0.1%. If the clone is mutated then it gives rise to a new ARB with *mortality* × 10 resources. If it is not mutated then the resources are returned to the parent ARB. The new clones are incorporated into the network and the processing of the data items continues.

4.5 THE ALGORITHM

Prior to the commencement of training the network an initial innoculation of ARBs must be provided. For the work presented here 10 ARBs were used to initialize the network for the trivial data set, and 30 ARBs were used for the iris data set. These numbers were used because they represent 20% of the number of items in each data set. The items from the data sets were simply every fifth one in whatever order they happened to be. Initial experimentation with different initial innoculations indicated no significant difference in behaviour when using different sub-sets of either data set.

Thus bringing all the above elements together, we can summarise the continuous algorithm as follows:

i) Innoculate the network with a random set of ARBs
ii) present a data item to all the nodes
iii) find the node with the highest activation
iv) allow this node to increase its resource level
v) deplete resources at all nodes
vi) cull nodes with less than threshold resource level
vii) allow highest activation node to clone
viii) relink the network with new clones
ix) return to ii)

5 EXPERIMENTS AND RESULTS

Results for two data sets in two different modes are presented. The first set of data consists of 50 two dimensional data items arranged in two clusters (see figure 8a). This was designed as a development tool to allow simple visualisation of ARB positioning in a well understood data set. The second set of data is Fisher's famous Iris data (see Fisher 1936) which provides a well

known benchmark data set with understood properties and some more challenging characteristics. The data consists of 150 four dimensional data items belonging to three categories, each of which represents a variety of Iris. A principal component plot (see Everitt 1974) of the first two principal components is presented in figure 8b. Both data sets were presented to the AIS as continuous streams of data which wrapped around each time the end of the data set was reached. The first two experiments were carried out using 20% of the data items as an initial innoculation and thereafter presenting all the data items from the outset. This type of analysis will be referred to from here on as *complete*. The last two experiments took one of the clusters from each data set and used 20% of this reduced set as an initial population and then trained for 250,000 data item presentations to demonstrate initial stability. Then the remainder of the data set was introduced and the network trained for a further 750,000 presentations to demonstrate the new stable state with the increased repertoire. This type of analysis will be referred to from here on as *incremental*.

5.1 COMPLETE ANALYSES

The complete analyses were carried out over 1,000,000 data item presentations to demonstrate long-term stability. The networks settle to a quasi-steady-state much more rapidly.

5.1.1 Trivial data

The networks produced for the complete analysis of the trivial data set very rapidly settled down to two distinct clusters of ARBs with the occasional appearance and disappearance of small outlying clusters or singlets which were rapidly culled (see figure 1). The network was examined at a large number of points during training and seemed to vary very little, although the addition and culling of clones occurred throughout (see figure 2).

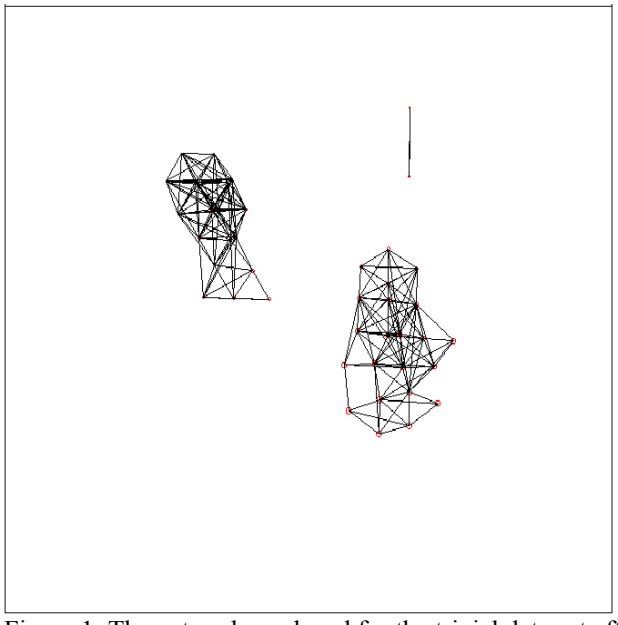


Figure 1: The network produced for the trivial data set after 30,000 data items have been presented

The size of the network settled down to between 40 and 55 quite rapidly. Variations in size and structure continued but did not vary the basic structure of the network after approximately 1000 data items had been presented and processed. Slight variations in size and structure are due to the stochastic nature of the network introduced by the cloning and mutation mechanism.

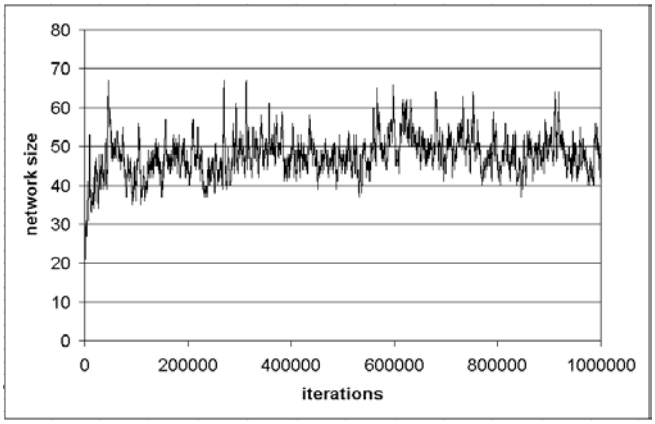


Figure 2: Size evolution of the network running on the trivial data set

The input space was densely populated in regions containing high densities of data throughout training. Regions of lower density outside the clusters of data were either devoid of ARBs, or supported small clusters of 1,2 or 3 ARBs for brief periods. These appeared due to the mutation of clones from the two groups.

5.1.2 Fisher‘s Iris data

This data set provides an interesting test for any data analysis technique as it consists of one clearly separable class of data (the Setosa class), and two slightly intermingled classes (the Virginica and Versicolor classes). A conventional Principal Component Analysis plot of the data shows this quite clearly in figure 8b. The network produced by the SSAIS after 350,000 data item presentations is shown in Fig. 3.

The evolution of this network was allowed to run on for 1,000,000 data presentations in order to examine the long-term behaviour of the network. The shape of the network was examined at various points and after about 100,000 iterations there were no major alterations in structure with a separate group for the Setosa class and an elongated group for the other two classes.

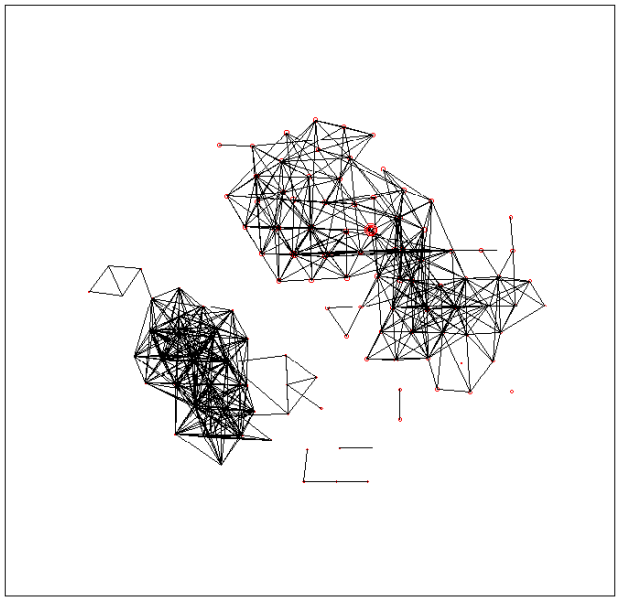


Figure 3: Network produced for the Iris data after 350,000 data item presentations

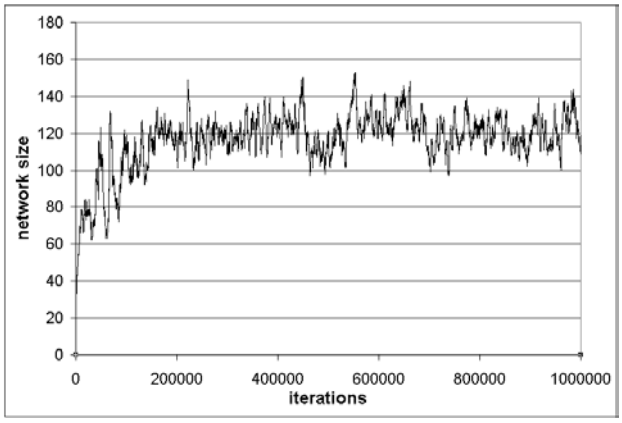


Figure 4: Size evolution of the network running on Fisher's Iris data set.

The long-term evolution of the size of this network is shown in Figure 4. The trace shows very rapid growth initially followed by gradual growth until about 150,000 iterations. Thereafter the network has a relatively stable size that varies by about 20 nodes either side of 120. This steady but dynamic behaviour is desirable as it indicates continuing introduction and maintenance of diversity within the network, whilst retaining reasonable coverage of the data space over a very long period. The enduring shape of the network can be seen in figure 5 which shows the final state of the network after 1,000,000 iterations.

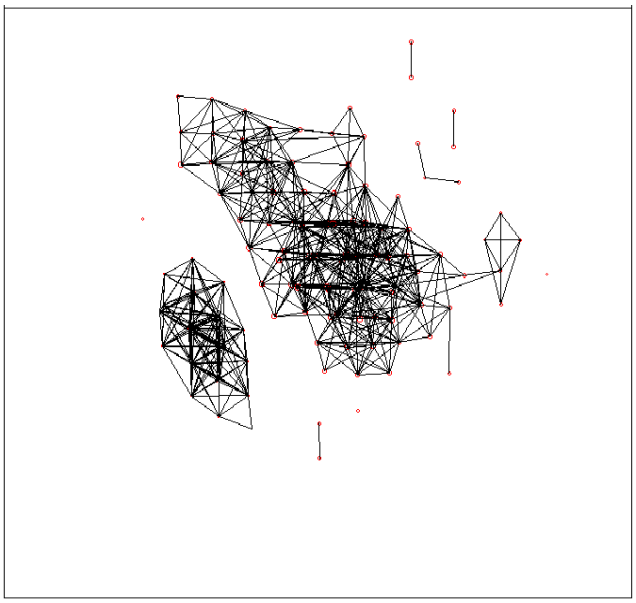


Figure 5: Network produced for Iris data after 1,000,000 data items have been presented.

Thus the networks produced throughout training on the Iris data cover the data space well, and reflect the nature of the groupings in the data.

5.2 INCREMENTAL ANALYSES

The incremental analyses were carried out over 1,000,000 data presentations in order to demonstrate the stability of the networks in their new configurations. Typically the behaviour of the networks settles down much more rapidly than this.

5.2.1 Trivial data

For the incremental analysis of the trivial data set the network was initialized with 5 of the 25 data items from the cluster close to the origin (see figure 8). The network was then trained for 250,000 data presentations with the members of only that cluster. The data being presented was then expanded to include the second group of data which is centred around the point (0.8,0.8). The size evolution of the network is shown in figure 6.

The network size can be seen to stabilise at the beginning of training at a size of between 30 and 45 nodes whilst only the first cluster of data is being used. The second cluster of data is introduced after 250,000 iterations after which the network takes about 200,000 more iterations to begin to cover the new data cluster. Examination of the intermediate networks produced shows little development of the network into regions which cover the new data. This seems to be due to the relatively confined region which the network covers before the second cluster is introduced. This lack of diversity in the network makes it unlikely that any mutated clone with only a single mutated antigen will be close enough to the new data items to survive. Thus the chance generation of several clones into the same region is required in order for the colonisation of the newly populated region

of input space to begin. Once a start has been made, the new region is rapidly covered quite effectively. This is shown by the increase in population size at 500,000 iterations. See figure 9 for network evolution.

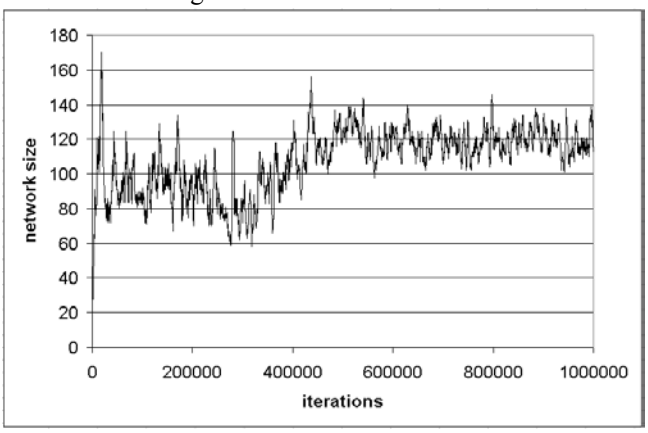


Figure 6: Size evolution of the network running on the trivial data set with introduction of second cluster at 250,000 iterations.

5.2.2 Fisher's Iris data

For the incremental analysis of Fisher's Iris data the network was initialized with 10 of the 50 Setosa class (see figure 8). The network was then trained for 250,000 data presentations with members of only that cluster. The data being presented was then expanded to include the other classes of data (Virginicas and Versicolors) which form a clearly distinct cluster. The size evolution of the network is shown in figure 7.

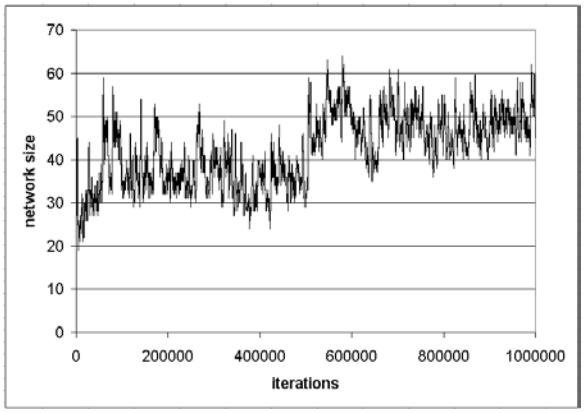


Figure 7: Evolution of network size for incremental analysis of Fisher's Iris data.

The network can be seen to have settled to a reasonably constant size of between about 70 and 110 when training on only the Setosa cluster (before 250,000 iterations). Subsequent to the introduction of the second cluster of data the network undergoes some fairly rapid changes. Initially there is a short period (between 250,000 and 300,000 iterations) of decline in size of the network. Then there is a period of quite rapid growth until about 450,000 iterations after which the network settles down to a fairly steady size of between about 105 and 130 nodes. Prior to the introduction of the second group of data the network consists of a single highly connected cluster of nodes. Upon the introduction of the additional data the network spreads out into a more complex structure before several chunks split off from the initial cluster and reform into a second large highly connected cluster. The ultimate shape which the network assumes is very similar to that produced by the complete analysis presented in section 5.1.2 (see figure 5). Snapshots of the network evolution throughout the incremental analysis are shown in figure 10.

6 DISCUSSION

The goal of this work was to create a genuinely stable, adaptive and continuous AIS. The changes that were introduced grew out of the realization that the shortcomings of the RLAIS (Timmis et al. 2001) stem from two fundamental problems: the nature of the resource allocation mechanism and the explicitly non-continuous nature of the epoch based update mechanism. The latter problem of assuming that there was an obvious point at which to stop presenting data items and perform an "update" was very simple to deal with. This just involved re-examining the algorithm and making sure that every operation could be carried out after the presentation of every data item. Most of the components of the system lent themselves readily to this approach, and as the resource allocation scheme was under scrutiny, problems with that aspect and the closely related problem of when and how much to clone were redesigned to fit the new regime. Successfully altering the resource allocation scheme required a little more thought. The fields of genetic algorithms and artificial life have taught many lessons about the nature of emergent behaviour in such systems, one of the most basic being that decentralization of control mechanisms usually leads to more interesting behaviour (see Johnson 2001). This led to the (now obvious) idea of devolving resource allocation to the ARBs, and adjusting the stimulation function to facilitate this. Thus now the only centralized function is that of choosing the winning ARB from the network. Finding the winner locally in the network would probably be possible, but unnecessarily complex and somewhat pedantic, especially as it could be argued that the bone marrow is a centralized controller of some importance in the biological immune system. Other mechanisms which allocate resources based on "local" winners were briefly examined and may be the subject of further research.

The time lag between the introduction of a new region of input data and the network covering the new region of the data space is disappointing. This is especially evident in the incremental analysis of the trivial data set. It seems clear that this lag is primarily due to a lack of diversity in the network. The network is slow to regain the diversity required to cover the new region due to the mutation and cloning mechanism, which is likely to produce mutations with only one data field different from the parent ARB. Thus it seems that examining more effective cloning and mutation mechanisms for the primary response would be of great interest. These are likely to involve an *artificial bone*

marrow that produces random antibodies when a poorly recognized pathogen is detected.

Control of the size of the network is to some degree removed from the domain of the user of the SSAIS, but clearly not entirely. The number of ARBs with which the network is initialized provides an initial point from which the system can evolve and thus provides a short-term control although the *mortality* constant and *decay rate* are far more sensitive and control the long-term meta-dynamics of the networks. The *mortality* constant provides a very coarse control which is unlikely to be changed in practice. The *decay rate* however provides a much finer control over the size of networks produced. Precisely how the size of the network relates to the *decay rate* will vary depending on at least the density of the data points in the input space, and the frequency of repetition of similar items. With fixed data sets the latter of these is simply the number of items in the set. The former is hard to measure, and its effect harder still. Some type of automatic and dynamic control of the *decay rate* would be extremely useful and remove a potential fudge factor.

7 FUTURE WORK

A number of pieces of work will flow directly from this approach to the construction of artificial immune networks:

i) The testing of the algorithm on some more complex data sets from the real world. This will enable some detailed comparisons with other techniques to be made, as well as to verify that the behaviour seen with the data sets presented here is repeatable.

ii) Running the algorithm on a continuously varying data source rather than fixed data sets presented many times to examine the flexibility of the representations formed and the rate at which the networks can track varying input.

iii) Creating an efficient and well engineered implementation of the algorithm. This will offer some performance increases, although performance has not proved to be a problem, as well as providing a stable software platform on which to base further experiments.

iv) Examining more realistic and intelligent cloning and mutation mechanisms. There is evidence that biological immune systems employ some very well controlled and directed cloning and mutation mechanisms, none of which are exploited here (see Kepler et al. 1993). Significantly different and potentially more useful behaviour could be expected if some methods such as these were applied.

8 CONCLUSIONS

The algorithm presented here generates networks of a bounded size over an indefinite number of data presentations and updates. The networks produced are continually changing whilst retaining good coverage of the input space and some diversity via the mutation mechanism employed. No central control in the form of a resource allocator is required which holds true to the distributed nature of the networks under construction. The system also has the advantage of being conceptually simpler than the previous resource limited artificial immune system. The dynamic stability displayed is a better model of the immune system than previous work presented and shows great promise for applications requiring analysis of continuously changing data sets with minimal intervention in the learning process.

Acknowledgments

Thanks to Jon Timmis for data, ideas and discussions.

References

B. Everitt (1974). Cluster Analysis, Heinemann, London.

J.D. Farmer, N.H. Packard (1986). The immune system, adaptation and machine learning, *Physica 22D*, 187-204.

R.A. Fisher (1936). The use of multiple measurements in taxonomic problems, *Annual Eugenics, Part II 7,* 179-188.

D.E. Goldberg (1989). Genetic algorithms in search optimization and machine learning, Addison Wesley.

N.K. Jerne (1974). Towards a network theory of the immune system, *Annals of Immunology 125C,* 373-389.

S. Johnson (2001). Emergence, the connected lives of ants, brains, cities and software, Penguin Press, London.

T.B. Kepler, A.S. Perelson (1993). Somatic hypermutation in B cells: an optimal control treatment. Journal of Theoretical Biology, 164, 37-64.

A. Perelson (1989). Immune network theory, *Immunological Review 110*, 5-36.

J. Timmis, M.Neal, J.Hunt (2000). An artificial immune system for data analysis, *Biosystems 55 (1/3),* Elsevier, 143-150.

J. Timmis, M.Neal, J.Hunt (1999). Data analysis with artificial immune systems, cluster analysis and kohonen networks: some comparisons. In *Proceedings of the International Conference on Systems Man and Cybernetics,* IEEE, Tokyo, Japan,. 922-927.

J. Timmis, M. Neal (2001). A resource limited artificial immune system for data analysis, *Knowledge-Based Systems 14,* Elsevier, 121-130.

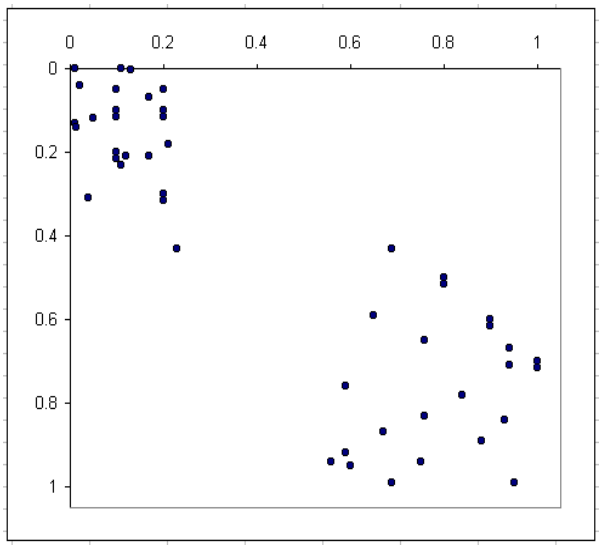
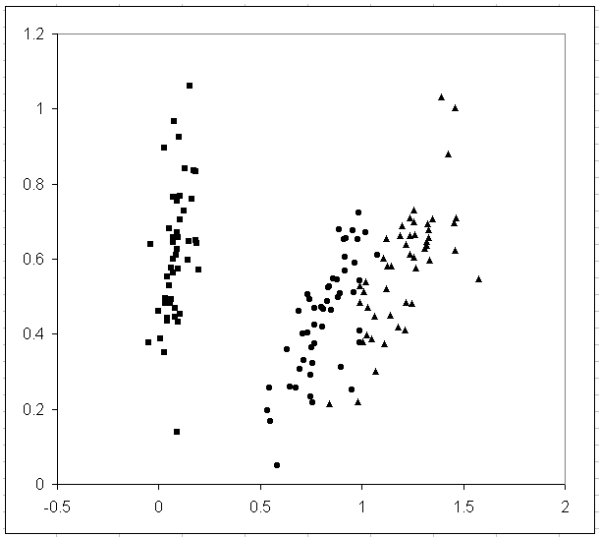
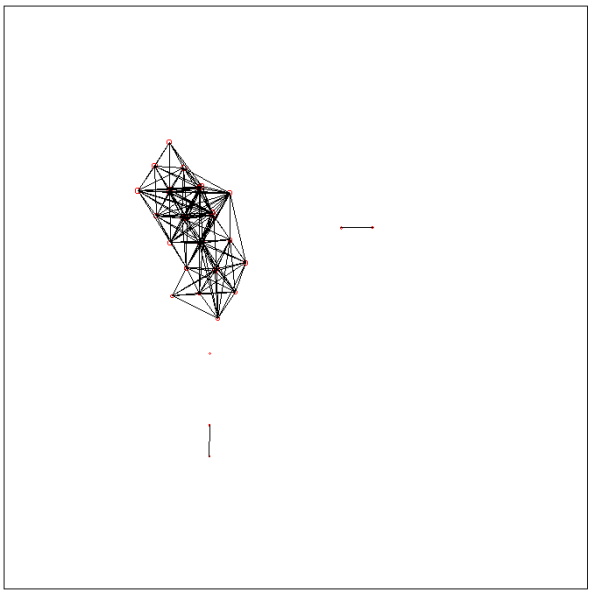
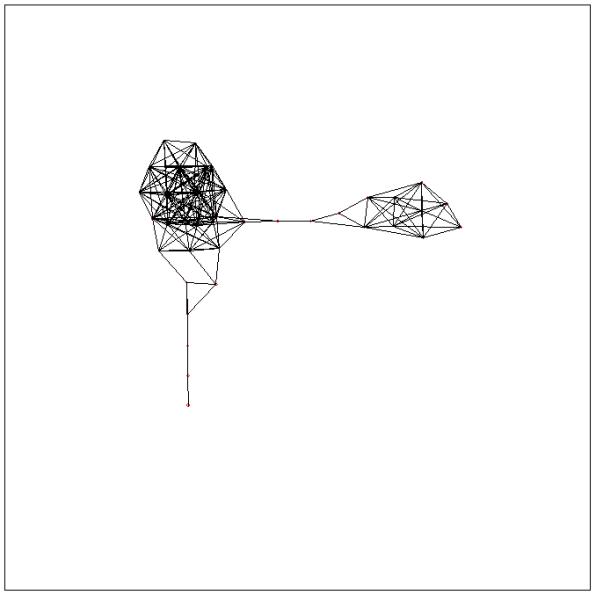


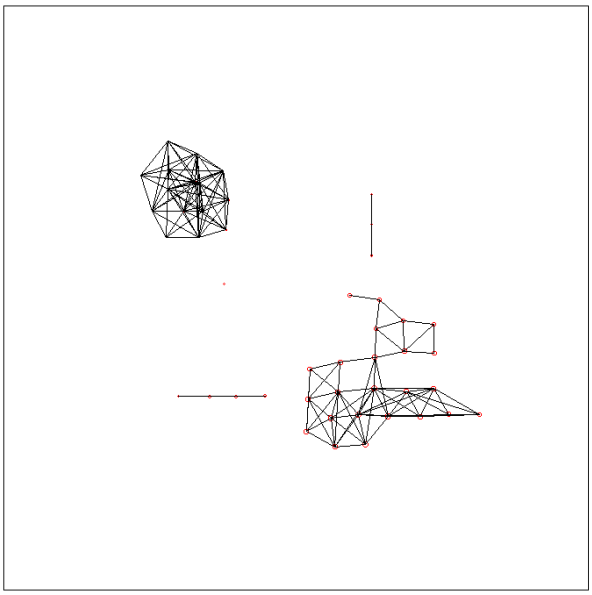
Figure 8: a) Two-dimensional trivial data set



b) Principal component plot of Fisher's Iris data. Setosa square, Virginica round, Versicolor triangular.







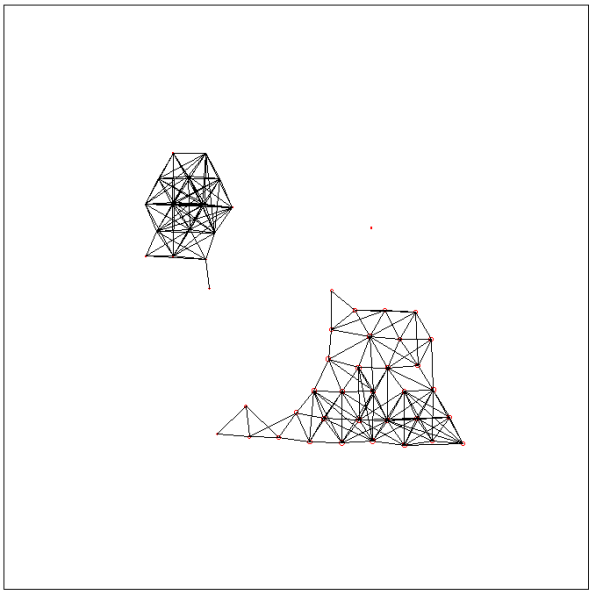
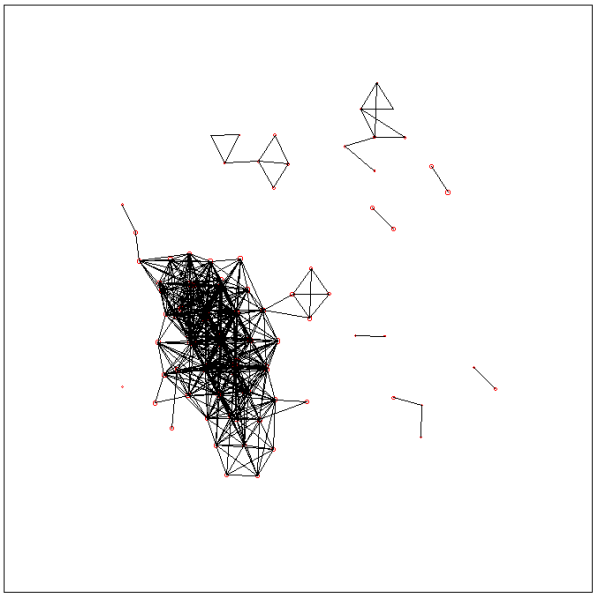
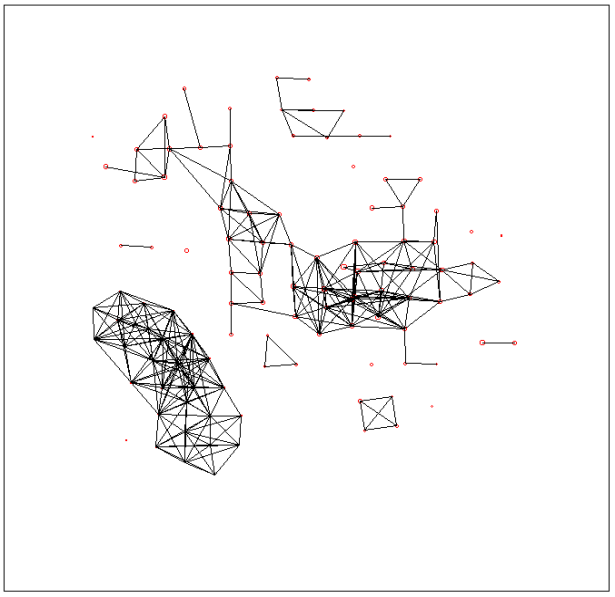
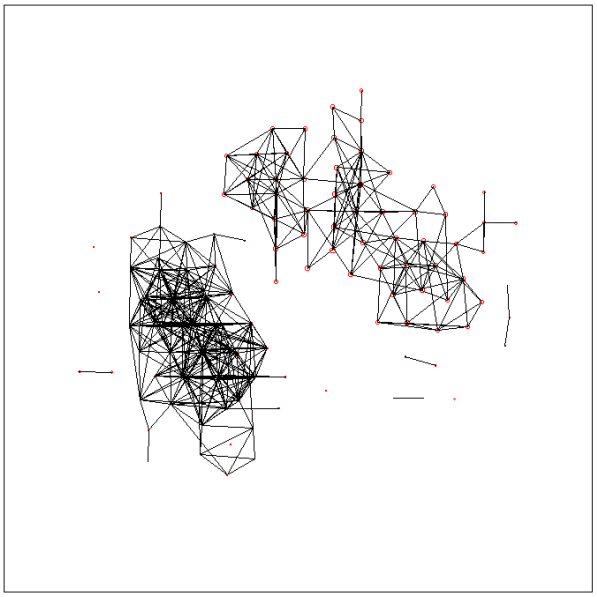


Figure 9: Network evolution during incremental learning of trivial data set. Series evolves top left to bottom right.

Shots taken at 250,000,300,000, 400,000 and 450,000







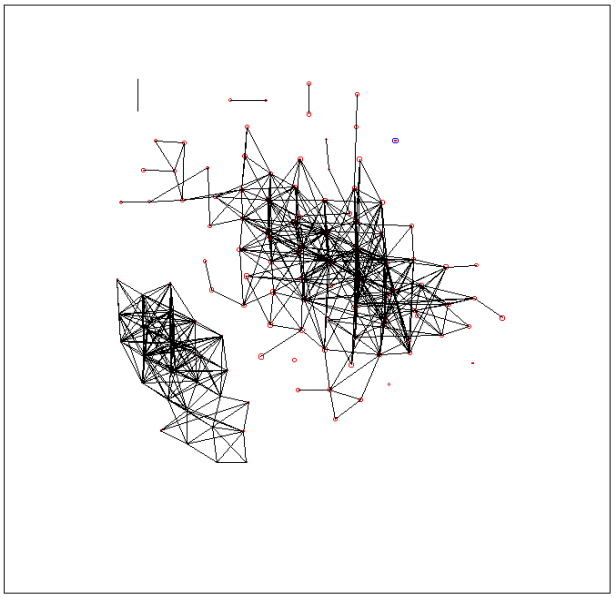


Figure 10: Network evolution during incremental learning of Fisher‘s Iris data. Series evolves top left to bottom right. Shots taken at 500,000, 550,000, 600,000 and 700,000

Negative Selection: How to Generate Detectors

Modupe Ayara, Jon Timmis, Rogério de Lemos
Computing Laboratory
University of Kent at Canterbury,
CT2 7NF, U.K.
{moa2, j.timmis,
r.delemos}@ukc.ac.uk

Leandro N. de Castro

Department of Computer and Electrical Engineering
State University of Campinas, Brazil.
lnunes@dca.fee.unciamp.br

Ross Duncan

Advanced Technology and Research Group
Self Service Strategic Solutions
NCR FSG Ltd., U.K.
ross.duncan@scotland.ncr.com

Abstract

The immune system is a remarkable and complex natural system, which has been shown to be of interest to computer scientists and engineers alike. This paper reports an on-going investigation into the usefulness of the negative selection metaphor for immune inspired fault tolerance. Various procedures to generate detectors for the negative selection algorithm are reviewed and compared in terms of time and space complexity for the production of competent detectors. A new algorithm has been identified and implemented. Experimentation was undertaken, and an analysis is presented on the effectiveness of the various algorithms. The outcome of this empirical analysis reveals that trade-offs have to be made in the choice of algorithm based on the time and space complexities, as well as the detection rate.

1. INTRODUCTION

As engineering and computing problems grow ever more complex, alternative sources of inspiration for solutions to these problems are being sought by computer scientists and engineers. Biology has been seen as a fruitful resource of inspiration with the creation of various biologically inspired techniques such as genetic algorithms, neural networks, and swarm systems (Bentley 2001). The immune system is now receiving more attention and is slowly being realized as a new biologically inspired computational intelligence approach (de Castro and Timmis 2002). An intuitive application of the immune system, and one that many researchers have followed, is to create artificial systems that have the ability to differentiate between self and non-self states: where *self* could be defined as many things, such as, normal behavior, normal network traffic between computers, and so on.

The next section explores one way in which the immune system allows for self non-self discrimination (negative selection), and reviews some approaches in artificial immune systems literature that have attempted to model this process. The main problems with these approaches are highlighted and a new algorithm has been implemented in an attempt to overcome some of these problems. The results presented in this paper demonstrate that the proposed algorithm is equivalent to the exhaustive algorithm for certain classes of problems, and even outperforms it in some cases for example clustered data. The fact still remains that none of the algorithms is able to resolve all the inherent problems associated with detector generation, thus some tradeoffs have to be considered when choosing an algorithm for generating detectors. The final section presents some conclusions and directions for future research.

2. USEFUL IMMUNOLOGY

The immune system is a remarkable and complex natural defense mechanism. The immune system responds to foreign invaders called *pathogens*. The first line of defense is known as innate immunity: this is the immune mechanism our bodies are born with (Janeway 1993). If the innate immune system cannot remove the pathogen, then the adaptive (or acquired) immune system takes over.

The adaptive immune system is made up of B and T-cells, which are capable of responding to certain antigenic patterns presented on the surface of pathogens. Receptors on B and T-cells match antigenic material and depending on the closeness of that match, T-cells stimulate B-cells into rapid proliferation and undergo affinity maturation.

Affinity maturation is a process by which stimulated B-cells are driven to become better tuned to the antigen responsible for initiating the immune response. This enhances the quality of the response (Staines, Brostoff et al. 1994). During affinity maturation, stimulated antibodies undergo a somatic mutation with high rates, termed *hypermutation*. The amount of mutation that a B-cell will undergo is inversely proportional to how well it matches the antigenic pattern: the higher the affinity (match) the lower the mutation, and vice versa. Production of antibodies from these B-cells then ensues, which ultimately remove the antigenic material.

Viewed from a computational perspective, this is an attractive learning mechanism and is one reason why the immune system has attracted such interest.

Pertinent to this work is the maturation of T-cells: what mechanisms are present to prevent the T-cells reacting against the own cells of the body? If this breakdown happens, it is known as an autoimmune disease. This is in part prevented via a process known as negative selection, that allows only the survival of those T-cells that do not recognize self cells. T-cells are produced in the bone marrow, but undergo a maturation process in the thymus gland, after which they are allowed to take part in an immune response. The maturation of the T-cells is conceptually very simple. T-cells are exposed to self-proteins. If this binding activates the T-cell, then the T-cell is killed, otherwise it is allowed into the repertoire. Cells that take part in an immune response are known as immunocompetent cells.

3. ARTIFICIAL IMMUNE SYSTEMS

Artificial immune systems (AIS) are adaptive systems inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving (de Castro and Timmis 2002). The important points of this definition are *inspiration* and *rationale*. In this case, the main idea is to develop problem solving tools that are inspired by the immune system. Through the use of the negative selection process described above, there have been a number of works attempting at building artificial immune systems for virus detection (Forrest, Perelson et al. 1994), computer security (Forrest, Hofmeyr et al. 1996), (Hofmeyr and Forrest 2000) and hardware fault tolerant systems (Bradley and Tyrell 2002). The original work by (Forrest, Perelson et al. 1994), in which the negative selection algorithm was proposed, has been inspirational to almost all the research in the AIS related to the computer security. More recently, that work has also provided the basis for building fault tolerant systems (Tyrell 1999). The basic idea of the algorithm is to produce a set of *change-detectors*, which can detect changes in what is considered normal behavior of a system.

4. NEGATIVE SELECTION: PRINCIPLES AND ISSUES

The negative selection algorithm is inspired by the maturation of T-cells in the thymus gland (Forrest, Perelson et al. 1994). The algorithm consists of two stages: *censoring* and *monitoring*. The censoring phase caters for the generation of change-detectors. Subsequently, the system being protected is monitored for changes using the detectors generated in the censoring stage. However, this algorithm is reported to be very time consuming (D'haeseleer, Forrest et al. 1996), (Wierzchoń 2000). The time taken to generate the detectors is measured by the number of candidate detectors that have to be examined before producing the required number of competent detectors. It was observed that the number of candidate detectors increases exponentially with the size of the self-set, at a fixed probability of not detecting non-self (Forrest, Perelson et al. 1994). This implies that the time to complete the process increases with the size of the self-set. Furthermore, this process does not check for redundant detectors. For minimizing these limitations, some variations of detector generating algorithm were developed: *linear* (D'haeseleer, Forrest et al. 1996), *greedy* (D'haeseleer, Forrest et al. 1996), and *binary template* (Wierzchoń 2000). Both the linear and greedy algorithms run in linear time respective to the size of the self and detector sets (D'haeseleer, Forrest et al. 1996). While the focus of the binary template is to generate efficient non-redundant detectors rather than minimizing the time to generate them. Work in (D'haeseleer, Forrest et al. 1996) claimed that the greedy algorithm manages to resolve this problem by generating a complete repertoire of detectors.

This paper includes the examination of time and space complexities of these algorithms, which were normalized for comparison. In order to cater for worst case situations, all the earlier assumptions included in the derivation of the original time and space complexities were discarded. For a more detailed comparison of several negative selection algorithms, please refer to (Ayara, Timmis et al. 2002).

4.1 EXHAUSTIVE DETECTOR GENERATING ALGORITHM

The exhaustive detector generating algorithm is the original method proposed by (Forrest, Perelson et al. 1994). The algorithm attempts to construct a set of *competent* detectors in the following way: (1) define the self data; (2) generate a random candidate detector; and (3) match each candidate detector generated with self data. If it matches with any self data, it is discarded, otherwise it is added to the collection of competent detectors. A flow diagram of the algorithm is presented in Figure 1 .

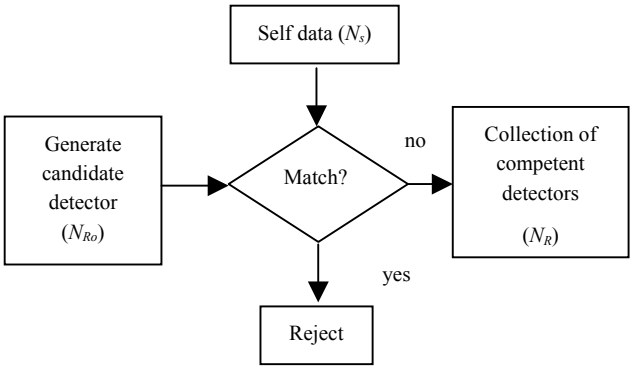


Figure 1: Exhaustive detector generating algorithm.

The time complexity of the algorithm was derived based on two factors: the time to generate a number of candidate detectors (N_{Ro}) and the time to compare each one of them with the population of self-data (Ns). The space complexity depends on the self-population, whose individual members are of length l. In (D'haeseleer, Forrest et al. 1996), the authors derived mathematical formulae to determine the computational complexities of the original algorithm. These were reviewed based on the following considerations: (1) generalising alphabet size m from binary {0,1}, where $m = 2$; and (2) the total number of candidate detectors (N_{Ro}) that can then be generated is m^l, where l is the length of each individual detector string.

Time and space complexities for this algorithm are presented in Section 7, while the empirical experiments carried out with the algorithm using 8-bits binary data, are presented in Section 6. The experiments confirm the limitation observed by (Forrest, Perelson et al. 1994) and (Kim and Bentley 2001), which is a costly computation of generating detectors.

The results motivated the examination and proposal of other algorithms to generate the set of candidate detectors. They are the linear, greedy and binary template algorithms. For the linear and binary template algorithms, please refer to (D'haeseleer, Forrest et al. 1996) and (Wierzchon 2000), respectively. The greedy algorithm will be analyzed in the following section due to its advantages of being linear in relation to the self-set as well as presenting a good coverage of non-self.

4.2 GREEDY DETECTOR GENERATING ALGORITHM

The greedy algorithm improves upon the linear algorithm through the elimination of redundant detectors. Furthermore, it ensures that generated detectors achieve as much coverage of non-self space as possible (D'haeseleer, Forrest et al. 1996). The algorithm is in two phases. The first is the processing phase taken from the linear algorithm, with the second phase being the actual process of generating detectors. This algorithm is based on the use of schemata proposed by (Helman and Forrest 1994) for the r-contiguous bits matching rule. The r-contiguous bits matching rule is a model of the affinity measure in the immune system. Assuming a binary representation of the self and detector strings, the r-contiguous bits matching rule compares a sequence of bits (of length r) in one string with a sequence of bits of the same length in the second string to see if they match. This approach, as shown in Figure 3, has been stated to closely capture the interaction between elements in the immune system (Percus, Percus et al. 1993). This is subject to a pre-defined matching threshold r that is the minimal length of contiguous bits strings common to the two strings for a match to occur. Given this matching rule, the schematic approach is to check for these common sub-strings, as depicted Figure 2, rather than the whole string.

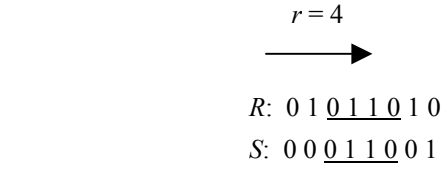


Figure 2: r-contiguous bits matching rule. The strings R and S of length $l = 8$, present $r = 4$ consecutive bits in common.

Assuming a matching threshold r, an alphabet size m (usually binary) and a length l, which is the length of each string, the first phase involves the generation of valid detector *templates* from a total number of m^r possible combinations. Templates are strings with r-contiguous significant bits that start from a specified bit position; and $l - r$ insignificant bits replaced with don't cares. Each template is constructed from a sequence of r bits that can be extended to fully specified detector strings. The set of valid templates are based on the self-data, such that only templates with no match in self are generated. These templates make up the first template array T_S where the nonzero entries constitute the valid templates.

During the second phase, detectors are generated through the extension of the templates to fully specified detector strings. After the generation of each detector string, all the templates that match the detector are removed from the set of valid templates for generating detectors. This prevents the generation of redundant and inefficient detectors at each step.

The time complexity of this algorithm depends on three factors: (1) the time to generate each valid detector templates in m^r; (2) the time to extend each valid template to a fully specified string; and (3) the time to update the templates T_R when creating each detector. The original time and space complexities were derived given these considerations (D'haeseleer, Forrest et al. 1996), but their corresponding mathematical formulae were derived based on the assumptions that each element of the template array can be evaluated in constant time. Also, the analysis ignored the earlier processing phases, before the valid number of detector templates are derived. Additionally, emphasis on binary alphabets can be extended to an alphabet size of m. These were incorporated into the reviewed formulae in Table 5.

5. NEGATIVE SELECTION WITH MUTATION

Work in (de Castro and Timmis 2002) proposed a slight modification of the exhaustive stage of the negative

selection, by introducing somatic hypermutation. Briefly, the procedure proposed the following: (1) define self data; (2) generate a candidate detector randomly; and (3) match each candidate detector with self data, if it matches, perform guided mutation on detector away from self. The guided mutation is performed on the candidate detector, which matches the self data. Mutation is then performed on the parts of the candidate detector that match with the self element. The mutation is adaptive, based on the affinity of the closely matching self element to a candidate detector. This means that the probability of mutation is directly proportional to affinity. Thus, the greater the affinity, the higher is the mutation probability. This idea was taken from the affinity maturation process of B-cells to antigenic patterns in the immune system. In this algorithm, however, the reference is the self-set, instead of non-self set. Hence, the mutation is performed proportionally to affinity to self-set, such that the candidate detector is changed so as not to match self-set. Also, this mutation approach was further augmented by the introduction of a life time indicator for a candidate detector. This in effect restricted the number of times mutation is performed on a candidate detector before a non-improved mutant is discarded. It was thought to have the desired effect of reducing the search space and hence, the number of candidate detectors generated.

The time complexity of NSMutation depends on the following factors: (1) the time to generate a random detector (each of length l) and compare with the population of self data to determine if they match; (2) assuming the use of r-contiguous bits matching, time to mutate matching region of length r in random detector away from self; and (3) a check for redundant detectors. In the worst case, m^l possible detectors can be generated when an alphabet size m is assumed. Hence N_{Ro} candidate detectors are equivalent to m^l. Also, mutation is limited to a region of length r in the candidate detector, which gives the upper bound of mutating the candidate detector as m^r. Subject to these factors, the time and space complexities are given in Table 5.

6. EXPERIMENTS

In order to verify the claims made in (Forrest, Perelson et al. 1994) and (Kim and Bentley 2001) with regards to the exhaustive algorithm, and additionally to test the efficacy of the proposed algorithms, experiments were undertaken using an 8-bit binary data test set. The exhaustive algorithm was used as the empirical standard for the experiments.

6.1 EXPERIMENTAL SETUP

The experiments were performed with randomly generated 8-bits data, with the inclusion of relevant parameters. The following subsections describe the procedures carried out for experimental set up.

6.1.1 Generating self data

As earlier stated, the 8-bit data used were randomly generated. The pseudorandom number generator of the Java 2 Platform (Standard Edition version 1.3) API was used to generate integer numbers between 0 and 255, which were then converted to 8-bit binary strings. During the experiments, there was a need to generate different sizes of self set. This was carried out by creating separate files for different population sizes of self sets.

6.1.2 Setting the matching threshold

The affinity between these binary strings (for the self-set, detector set and test data) was determined using the r-contiguous bits matching rule. The optimal value for matching threshold (r) had to be obtained by changing values of r from 1 to l. This process was done in order to obtain the combined values of correct and incorrect classification by detectors generated using a specific threshold. Correct classification value is derived from the sum of true positive (rate at which non-self is correctly detected) and true negative (rate at which self is correctly not detected). While incorrect classification is the sum of false positive (rate at which self is incorrectly detected) and false negative (rate at which non-self is not detected). Both the correct and incorrect classification values are used to determine the appropriate values of r. This is different from the approach used by (Kim and Bentley 2001) as well as the suggested method in (D'haeseleer, Forrest et al. 1996). In (Kim and Bentley 2001), the value of r was determined from the equations in (Forrest, Perelson et al. 1994), which yielded poor values of matching threshold for the corresponding data. While (D'haeseleer, Forrest et al. 1996) proposed an approach based on the greedy algorithm. Both approaches reveal that there is no hard-and-fast rule for setting this parameter, rather various values can be tested in order to select the optimal one. The following procedure was carried out to determine this parameter:

1. Generate self and test sets from the data sets being experimented upon;
2. Generate required detectors N_R (using equation: $N_R = \frac{-\ln(P_f)}{P_m}$) for different values of r which are varied from 1 to l;
3. Test the detectors generated on the test file to obtain their correct and incorrect classification rates;
4. Use the value of r for which there is minimal incorrect classification and maximum correct classification rates in subsequent experiments.

An outcome of the procedure is illustrated in Table 1 based on the mean values of correct and incorrect classification rates obtained over 10 trials using the following parameters: self set $N_S = 8$, test set $N_T = 256$, available (N_R), and potential (N_{Ro}) repertoires. Given this table, a matching threshold value of 8 will be preferable to the other values since it yields maximum correct and minimum classification rates. When the matching threshold was set to values below 3, no detectors could be generated. This indicates that at such threshold values, all the detectors match the strings in the self-set. The value of *r* thus determines the proper partitioning of the data space into self and non-self segments. This makes the choice of an optimal value for *r* crucial to the effectiveness of the change-detection function.

Table 1: Test for obtaining optimal value of matching threshold (*r*)

r	N_{Ro}	N_R	Correct classification rates	Incorrect classification rates
3	209.2	5	41.80%	58.20%
4	37.30	12	56.02%	43.98%
5	46.50	29	70.98%	29.02%
6	87.90	74	85.12%	14.88%
7	210.90	196	89.10%	10.90%
8	604.80	589	91.72%	8.28%

6.1.3 Mutation probability

The mutation probability *(mutProb)* is a threshold that determines whether a bit position in a binary string will be mutated or not. This value was initially implemented using an adaptive mechanism which is calculated as the length of the matching bits in two binary strings divided by the length *l* of the binary string. The value generated is a real number between 0.0 and 1.0. This threshold value is then used to determine whether a bit position is subjected to mutation. For each bit position to be mutated, if a randomly generated number between 0.0 and 1.0 is less than the mutation probability, the bit is mutated. The converse is the case when the random number is greater than the mutation probability. In (Ayara, Timmis et al. 2002), the adaptive mutation probability was discovered to degrade the time complexity of the algorithm if the probability is greater than a specific value. This is because the probability indicates that a sizeable fraction of the total number of bits in a random binary string matches self. Hence the process of mutating a random detector is restricted to limited options. This can be explained by a matching threshold $r = 8$. In this case, the mutation probability is 1 and the process of mutation just flips a random detector to its image. In a situation that the image also matches self, mutation flips back to the original detector which also matches self. If this is the case for a significant number of random detectors generated, the time complexity is increased considerably. However there is a threshold value below which this will not occur. For example, the results of experiments in (Ayara, Timmis et al. 2002) show that using 8-bits binary data generated randomly the maximum mutation probability that will not make the algorithm worse off than the original exhaustive, for threshold values of 7 and 8, was confirmed to be 0.8. This directed the choice of mutation probability for subsequent experiments, which was set to 0.5.

6.1.4 Detector life-time indicator

The detector life-time indicator (*mutLim*) determines the number of attempts that mutation can be performed on a random detector. When values of this parameter are greater than 1, it was found to increase the time complexity of NSMutation when used with adaptive mutation probability. This phenomenon can be linked to the explanation given in section 6.1.3, which accounts for the poor behaviour of the algorithm using adaptive mutation probability. In a situation when the mutation probability is above a specific value, and the limited detector options that mutation can generate also match with self, an increase in life-time indicator only extends the time for the flipping the detector back and forth between the image and the original detector.

Some definitions of terms used in the experiments are listed in Table 2.

Table 2: Definitions of terms used in experiments

Terms	Definitions
l	Length of string
r	Matching threshold
m	Alphabet size
N_S	Population of self data
N_{Ro}	Population of candidate detectors
N_R	Population of competent detectors
P_m	Probability of detecting a non-self
P_f	Probability of failing to detect non-self
N_T	Population of test data
mutProb	Mutation probability
mutLim	Mutation limit (Detector life-time indicator)

6.2 THE BOTTLENECK FOR NEGATIVE SELECTION

Given the earlier discussion regarding the constraint of the exhaustive algorithm, i.e., the size of the set of candidate detectors increases exponentially with the size of the self-set, initial tests were performed to check if this claim holds true for the proposed algorithm. This process involved determining the number of candidate detectors required to produce a specified number of competent detectors when the population size of self is increased progressively. The test was carried out with both the NSMutation and exhaustive algorithm for comparison.

Using the definitions provided in Table 2, the mathematical equations for estimating P_m, N_R, and N_{Ro} (Forrest, Perelson et al. 1994), were employed for implementing the algorithm.

The following procedures were carried out for NSMutation algorithm:

1. For a particular data set, derive r (section 6.1.2) for all runs of the experiment;
2. Calculate P_m and select a desired value for P_f ;
3. Determine the value of N_R according to the following equation: $N_R = \frac{-\ln(P_f)}{P_m}$;
4. Set the values of *mutProb* and *mutLim* using the guidelines in sections 6.1.3 and 6.1.4 respectively;
5. Execute steps a-c a number of times while incrementing the size of N_S, $8 \le N_S \le 160$. (The selected value was 100 for trial runs):
    a. Determine N_{Ro} experimentally by generating random strings until N_R valid detectors are determined;
    b. Once a match occurs between a self string and a candidate detector, or there is a duplicate of the detector in the detector set, perform uniform mutation in a guided manner until the candidate detector becomes a competent detector. The detector is then added to the set of useful detectors;
    c. The number of mutation attempts is limited by a detector life time indicator (*mutLim*), which is set to a fixed value.

This life-time indicator constrains the time expended to change a detector that closely resembles self. In a situation where a mutated detector is not improved by the time the life-time has expired, it is discarded and replaced by another random detector. The same process was undertaken for the exhaustive algorithm, excluding the mutation operator and the check for redundant detector in the detector set. The *potential repertoire size* (N_{Ro} - collection of candidate detectors before negative selection) for both algorithms was recorded for comparison. While the population of detectors generated after negative selection, known as the *available repertoire size* (N_R), was set as a parameter for the simulation. The results obtained from the experiments are presented in Table 3 and Figure 3. These results are obtained from 100 trials for each size of the self-set, $8 \le N_S \le 160$, with the following parameters $N_R = 589$, $r = 8$, $P_f = 0.1$, *mutLim* = 4, *mutProb* = 0.5. Each column of Table 3 holds values calculated as a mean of the number of trials, while the standard deviations are enclosed in brackets for each mean value. Column (a) indicates the size of self set, (b) holds the theoretical estimates of potential repertoire (N_{Ro}), (c) the experimental N_{Ro} values for the exhaustive algorithm, (d) experimental N_{Ro} values for NSMutation, and (e) the mean mutation occurrence over 100 trials.

The results in Table 3 are selected from the outcome of the experiments shown in Figure 3. From Table 3, it can be clearly seen that the potential repertoire generated for both algorithms are similar, for example when the population size of self set is 152, the exhaustive and NSMutation algorithms generate potential repertoire of 1128.16 and 1127.62 respectively. This explains the overlap in the graphs of both algorithms. Also, column (e) in Table 3 show that mutation occurs 1.807 ≈ 2 times out of 100 trials.

In order to determine the effectiveness of the NSMutation in comparison to the exhaustive algorithm, their detection rates were tested empirically using a single population size of self $N_S = 8$. Other parameter values include $N_R = 589$, $r = 8$, $P_f = 0.1$, *mutLim* = 4, *mutProb* = 0.5, $N_T = 256$. The outcomes of these tests are presented in Table 4.

As shown in Table 4, the theoretical estimation of potential repertoire size is calculated as 608.21, while the mean potential repertoire sizes for exhaustive and NSMutation respectively are 608.10 and 608.40. Their corresponding detection rates are 90.36% for exhaustive and 89.84% for NSMutation. Testing the statistical difference between their detection rates using the Z-test, gave a value of +0.085, which shows that their detection rates are not statistically different.

Table 3: Experimental results generated from 8-bits data based on 100 trials for self set N_S = 152, 160.

N_S	N_{Ro} (Theoretical)	N_{Ro} (Exhaustive algorithm)	N_{Ro} (NSMutation algorithm)	Mutation Occurrence
(a)	(b)	(c)	(d)	(e)
152	1068.62	1128.16 (33.130)	1127.62 (31.739)	1.807 (1.020)
160	1102.61	1084.26 (32.344)	1091.14 (31.190)	1.768(0.992)

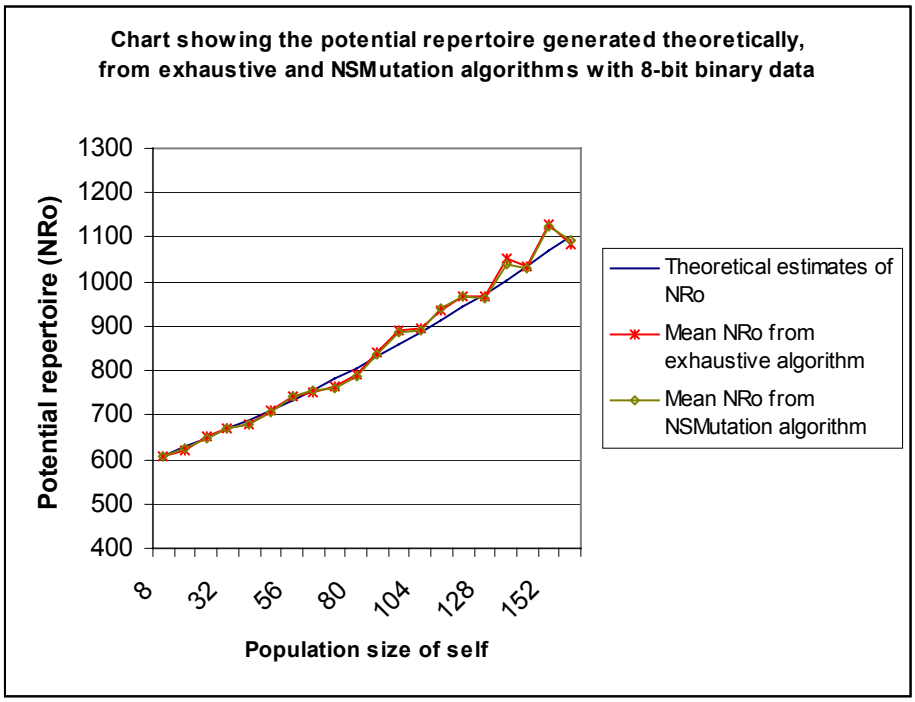


Figure 3: Chart for 8-bits data that illustrates the theoretical estimates and mean population of potential repertoire based on 100 trial runs for both algorithms given each size of self-set.

Table 4: Test results of detection performance of the NSMutation and exhaustive algorithms over 10 trials.

	Exhaustive	NSMutation
Theoretical N_{Ro}	608.21	
Experimental N_{Ro}	608.10	608.40
Detection rates	90.36%	89.84%
Z-test value	+0.085	

7. ANALYSIS AND DISCUSSION

In this section, the output of the experiments performed in section 6.2 are analyzed with the aim of discussing the features peculiar to NSMutation and comparing with the exhaustive algorithm in terms of tests conducted. This comparison is then extended to the other algorithms for an overview of all the detector generating algorithms.

From the diagram of Figure 3, it can be observed that the number of candidate detectors examined for the exhaustive algorithm increases exponentially with the size of the self-set. This confirms the limitation expressed by (Kim and Bentley 2001). This behavior is also exhibited by NSMutation, whose pattern of increase in potential repertoire closely resembles that of the exhaustive algorithm. This can be explained to be a result of the random nature of the self set, which is normally distributed. During the process of mutating a candidate detector for the NSMutation algorithm, the aim is to guide the candidate detector away from self set. But since the self set is randomly distributed around the search space, there is an equal probability of mutating the random detector away from or towards self set. Hence the impact of guided mutation cannot be guaranteed for random data, and the outcome is more or less a random generation of detectors. However, this is not the usual case for a clustered self set with well-defined boundaries. Preliminary experiments performed in (Ayara, Timmis et al. 2002) to test this showed that the potential repertoire is almost linear with increase in the self set. Also refer to (Ayara, Timmis et al. 2002) for the pseudocodes of all the algorithms.

The comparison of their detection rates in Table 4 further confirms the similarities. The difference in their performance at detecting non-self was evaluated using the Z-statistic at a significance level of 0.05%, and the outcome showed that their detection rate performances were not statistically different.

Although from Figure 3 it can be asserted that the NSMutation algorithm behaves similarly to the exhaustive, some extensive studies of the NSMutation algorithm (Ayara, Timmis et al. 2002), provide more information about some parameters of the algorithm that control its performance. They include the matching threshold (*r),* detector life-time rate (*mutLim*) and mutation probability (*mutProb*). These parameters can deteriorate its performance than its predecessor or speculatively better, if a good combination of parameters for the data set can be obtained. For example, when $r = l$ (length of each string), the effect of the *mutProb* on the time complexity is more profound, even though there is a higher chance of generating good detectors due to the exact matching. The effect of *mutProb* is aggravated by a high value of *mutLim*. For example when $l = 8$, $r = 8$, *mutLim* = 4, and *mutProb* = 1.0, mutation of a non-competent detector produces its image and if the mutant also matches self, further mutation just flips the image back to the original detector, thereby causing an alternation between the image and the original detector. The *mutLim* parameter thus causes this process to be carried out for a specific number of trials. However, as $r << l$, the effect of *mutProb* and *mutLim* pale into insignificance, since the value of r already triggers high time complexity. This parameterization factor for good performance of learning algorithms has been observed by (Bentley, Gordon T. et al. 2001). So the next question to be answered is "what parameter values for the NSMutation algorithm can make it outperform the exhaustive?"

Altogether, the reviewed and normalized time and space complexities of all the algorithms, as shown in Table 5, reveal the characteristics in terms of computational complexity. While the time complexities of the exhaustive algorithm and NSMutation are exponential with respect to the size of self, the others have time complexities that are linear functions of the self. The linear algorithm, however, has the disadvantage of generating redundant detectors, as is the case with the exhaustive; this in turn limits its performance. However, the greedy algorithm achieves the best coverage for detection, due to the fact that it generates complete repertoires of detectors as claimed by (D'haeseleer, Forrest et al. 1996). The binary template, which derives its inspiration from the greedy also achieves similar coverage. Both greedy and binary template algorithms have higher computational complexity when compared to the linear algorithm. The greedy algorithm includes the process of checking that each detector generated represents a cluster of non-self to prevent redundancy and also ensure that efficient detectors are produced. Also the binary template algorithm includes similar processes of removing redundant detectors and ensuring that inefficient detectors are eliminated. Hence, these additional processes of guaranteeing non-self coverage and non-redundancy incur extra time to complete the algorithms. It must be noted that when the matching threshold r approaches length l of each string in the search space, the linear time complexities of the linear, greedy and binary template with respect to the size of the self-set, may exhibit similar behavior as that of the exhaustive and NSMutation, due to the exponential value m^r in their time complexity equation.

In terms of space complexity, NSMutation has a higher space complexity that the exhaustive. The reason for this is that the NSMutation stores the detectors as they are generated for comparison with subsequent detectors in order to prevent redundancy. On the other hand, the linear, greedy and binary template incur more space complexity due to the storage of m^r binary template strings that are stored and updated. However the binary template algorithm has a lower space complexity when compared with the linear and greedy algorithms.

Another criterion for comparing the algorithms is the coverage of detectors. This factor measures the extent to which the detectors generated from the negative selection algorithm are fully representative of the non-self set. Thus it thereby provides a means of determining the efficiency of the algorithm. If complete coverage is to be achieved, it implies that all non-self detectors must be generated. However, there is a need to maintain a balance between the time taken to generate detectors and getting a good coverage. This balance seems to be best achieved by the greedy

algorithm. The algorithm is able to generate non-redundant detectors that have high detection coverage, at minimal time complexity.

In summary, it can be argued that the NSMutation is more or else the exhaustive algorithm since they expend similar time complexity and achieve as much coverage of non-self. However, NSMutation differs from the exhaustive algorithm because it includes checks for redundancy and tunable parameters that can induce different performance. When compared with the linear, greedy and binary templates, the simplicity of NSMutation makes it quite attractive as against the others that entail cumbersome procedures. Furthermore, only the exhaustive and NSMutation can be used with other matching rules. The linear, greedy and binary template algorithms are restrictive. They are limited to the r-contiguous bits matching rule, which renders them inextensible and inappropriate for other matching rules. The benefits of NSMutation thus include simplicity, high detection rate performance and extensibility.

Table 5: Reviewed time and space complexities of all detector generating algorithms (refer to original equations in (D'haeseleer, Forrest et al. 1996)).

Algorithm	Time	Space
Exhaustive	$O(m^l.N_S)$	$O(l.N_S)$
Linear	$O((l-r+1).N_s.m^r)+$ $O((l-r+1).m^r)+$ $O(l.\ N_R)$	$O((l-r+1)^2.m^r)$
Greedy	$O((l-r+1).N_s.m^r)+$ $O((l-r+1).m^r.N_R)$	$O((l-r+1)^2.m^r)$
Binary Template	$O(m^r.N_S)+$ $O((l-r+1).m^r.N_R)$	$O((l-r+1).m^r)+$ $O(N_R)$
NSMutation	$O(m^l.N_S)+O(N_R.m^r)$ $+O(N_R)$	$O(l.(N_S + N_R))$

8. CONCLUSIONS

This paper has made a comparison between the different negative selection algorithms for generating detectors, and implemented a variation of the initial exhaustive algorithm. The results were presented using the time taken to generate detectors, as well as the detection rate coverage of the final detectors generated. It has been demonstrated that there are trade-offs to be made in deciding on the best algorithm for producing the detectors. The exhaustive algorithm takes considerable time (exponential in size of self data) and produces redundant detectors; the linear algorithm has a linear time complexity but also produces redundant detectors; the greedy algorithm produces a complete repertoire using up as much space as the linear algorithm, but has a higher computational complexity; the binary template produces a minimal set of efficient detectors at the expense of more time complexity; and finally NSMutation is similar to the exhaustive algorithm with the difference of eliminating redundancy and possessing parameters that can be optimized for better performance. However for structured data sets, the NSMutation has shown better performance in terms of time complexity, but there is still need for further verification. Thus, in a case where choice has to be made between both exhaustive and NSMutation, the latter has the advantages of possessing tunable parameters, eliminating redundant detectors, and being suitable for any matching rule. But, the decision lies with the constraints being met while implementing the algorithm in its target domain. Different domains place emphasis on different constraints that must be satisfied. These might include factors such as time to generate detectors; space storage used by the detectors; matching function; as well as the performance of detectors generated. Since no algorithm has managed to minimize all these constraints, trade-offs have to be made in choosing an algorithm for generating negative selection detectors. But it must be said that more analysis of the NSMutation algorithm will need to be carried out in order to determine the best combination of parameters that can improve it significantly.

Acknowledgments

We would like to thank NCR FSG for their continued financial support and valuable input into this research. Leandro N. de Castro would like to thank CNPq (Profix 540396/01-0) for their financial support.

We gratefully acknowledge comments received from anonymous reviewers.

References

M. Ayara, J. Timmis, et al. (2002). An Investigation into Negative Selection for Change Detector Generation, *Technical Report*, Computing Laboratory, University of Kent at Canterbury, U.K.

D. W. Bradley and A. M. Tyrrell (2002). A Hardware Immune System for Benchmark State Machine Error Detection. *Congress on Evolutionary Computation. Part of the World Congress on Computational Intelligence*, 813-818, Honolulu, HI. USA, IEEE,.

P.J. Bentley. (2001) *Digital Biology*. Hodder Headline.

P. J. Bentley, T. Gordon, J. Kim and S. Kumar (2001). New Trends in Evolutionary Computation. T*he Congress on Evolutionary Computation (CEC-2001)*, 162-169, Seoul, Korea, May 27-30, 2001.

L. N. de Castro and J. I. Timmis (2002). Artificial Immune Systems: A New Computational Intelligence Approach, Springer-Verlag.

P. D'haeseleer (1996). An Immunological Approach to Change Detection: Theoretical Results. *IEEE Computer Security Foundations Workshop*, Dromquinna Manor, County Kerry, Ireland, June 10-12, 1996.

P. D'haeseleer, S. Forrest, et al. (1996). An Immunological Approach to Change Detection. In *Proc. of IEEE Symposium on Research in Security and Privacy*, Oakland, CA.

S. Forrest, S. A. Hofmeyr, et al. (1997). Computer Immunology, *Communications of the ACM*, 40(10):88-96.

S. Forrest, A. Perelson, et al. (1994). Self Nonself Discrimination in a Computer. In *Proc. of IEEE Symposium on Research in Security and Privacy*, 202-212, Oakland, CA, May 16-18, 1994.

P. Helman and S. Forrest(1994). An Efficient Algorithm for Generating Random Antibody Strings. *Technical Report CS-94-07*, The University of New Mexico, Albuquerque, NM.

S. Hofmeyr and S. Forrest (2000). Architecture for an Artificial Immune System. *Evolutionary Computation*, 7(1):45-68.

C. A. Janeway (1993). How the Immune System Recognizes Invaders. *Scientific American*: 41-47.

J. Kim and P. Bentley (2001). Investigating the roles of Negative Selection and Clonal Selection in an Artificial Immune System for Network Intrusion Detection, *Technical Report*, Dept. of Computer Science, UCL, London.

J. K. Percus, O. E. Percus, et al. (1993). Predicting the Size of T-cell Receptor and Antibody Combining Region from Consideration of Efficient Self-Nonself Discrimination. *National Academy of Science*, 90:1691-1695.

N. Staines, J. Brostoff, et al. (1994). Introducing Immunology, Mosby.

A. M. Tyrrell (1999). Computer Know Thy Self!: A Biological Way to Look at Fault-Tolerance. *Second Euromicro/IEEE Workshop on Dependable Computing Systems*, 129-135, Milan, Italy.

S. T. Wierzchon(2000a). Generating Optimal Repertoire of Antibody Strings in an Artificial Immune System. In M. Klopotek, M. Michalewicz and S. T. Wierzchon (eds.) *Intelligent Information Systems*. Advances in Soft Computing Series of Physica-Verlag/Springer Verlag, Heidelberg/New York 2000, Physica-Verlag, 119-133.

Anomaly detection using negative selection based on the r-contiguous matching rule

Shantanu Singh

DaimlerChrysler Research Centre India Pvt. Ltd.,
137 Infantry Road,
Bangalore 560001,
India.
shantanu@rti.daimlerchrysler.com

Abstract

A series of observations of a system over time is often used to characterize its normal behaviour. The problem of anomaly detection is that of finding deviations in the characteristics of the system. Anomaly detection algorithms inspired by the negative selection mechanism of the natural immune system have been proposed. This paper presents results obtained by employing an efficient negative selection algorithm based on the r-contiguous matching rule to detect anomaly in various forms of data. The algorithm presented is an extension of an existing detector generating algorithm to deal with *m*-ary alphabet strings. Results are obtained for three cases – assembler instructions, system calls and simulated time series. Finally, conclusions of the study are presented and future direction of the work, currently in progress, is indicated.

1 Introduction

The negative selection algorithm for anomaly detection is inspired by the way the natural immune system generates T-cells through a censoring process in the thymus. T-cells have receptors that bind with proteins. Only those that do not bind with self-proteins are released into the rest of the body[1]. These T-cells then perform the role of detecting non-self entities in the body by binding with them. To extend the same principle to other systems, Forrest *et al.* (1994) have proposed that the characteristics of the system can be expressed in finite alphabet strings, with self strings corresponding to normal behaviour and nonself strings corresponding to anomalous behaviour. The anomaly detectors that are generated would be those that match any string not among the self strings. This method of generating detectors is known as negative selection. In this paper, we use the r-contiguous matching rule to define a match. Two strings are said to have an r-contiguous match if they are identical in at least r contiguous positions. This partial matching rule is used since two strings of a reasonable length rarely have an exact match. Wierzchoń (2000) has investigated the discriminative ability of this matching rule.

Forrest *et al.* (1994) have presented a simple detector generating algorithm which could be used with any matching rule. D'haeseleer *et al.* (1996) have described more efficient algorithms based on the r-contiguous matching rule which run in linear time. The greedy detector generating algorithm (GDGA) (D'haeseleer *et al.* 1996) generates detectors which cover more non-self space than other algorithms that have been proposed. This algorithm has been defined for a binary alphabet.

In the present work, we introduce an algorithm that is an improvement over GDGA. This algorithm handles strings with higher alphabet size. It has the advantage of retaining the semantics of the information present in the strings. This is relevant in cases where a binary encoding of the string would be undesired. For example, in strings constructed from system call data, it is necessary to retain the uniqueness of the letters of the alphabet, where the alphabet is defined by the set of all system calls.

Section 2 explains the extended algorithm. Section 3 presents the results obtained by this algorithm on simulated time-series data, assembler instruction data and system call data. A discussion on some observations is presented in Section 4. Section 5 puts forth the conclusions. Future direction of work is indicated in Section 6.

2 Extension of GDGA for Higher Alphabet Size

The greedy detector generating algorithm (binary GDGA) presented in D'haeseleer *et al.* (1996) uses a greedy heuristic to achieve a better coverage of a nonself space. The algorithm has been defined for a binary alphabet. Here, an extension to the algorithm for a higher alphabet size is presented. The following notation will be used :
A is the alphabet of size m.
s is a string composed of symbols from A.
$\hat{s}$ is string s with its leftmost symbol removed.
$s.x$ is string s with $x \varepsilon A$ appended to it.

[1] The process T-cell maturation also involves positive selection, wherein only those T-cells that have an ability to interact with self major histocompatibility antigens are selected.

A *template* of order *r* is defined as a size *l* string consisting of *l-r* unspecified positions and *r* contiguous symbols (ε *A*). $t_{i,s}$ is a template with string *s* forming the fully specified positions starting at position *i*.
The right (left) *completion* of a template is a template string with the blank positions to the right (left) filled up with symbols from *A*.
Two strings have an *r*-contiguous match if they are identical in at least *r* contiguous positions. Similarly, a template matches a string if its fully specified positions are exactly matched at the same positions in the string.
Arrays of dimensions $m^r \cdot (l-r+1)$ are used for the creation of detectors. The rows of these arrays are indexed by strings composed of symbols from *A*. To get the numerical value of a string *s*, we give each symbol of *A* a unique integer value in [*0, m)* and then consider *s* to be a number expressed in base *m* number system.
Boolean arrays C_S and C_S' are based on the set of self strings S. Integer arrays C_R and C_R' are based on the detector set R.
$C_S[s][i]$ ($C_S'[s][i]$) is false if there are no right (left) completions of $t_{i,s}$ unmatched by any string in S and true otherwise. C_S can be calculated using the recurrence relationship of its elements in the following way:

$$C_S[s][l-r+1] = \begin{cases} false, \text{ if } t_{l-r+1,s} \text{ is matched in S} \\ true, \text{ otherwise.} \end{cases}$$

$$C_S[s][i] = \begin{cases} false, \text{ if } t_{i,s} \text{ is matched in S} \\ \bigvee_{x \in A} C_S[\hat{s}.x][i+1], \text{otherwise.} \end{cases}$$

C_S' can be computed using a similar recurrence relation as for C_S.
$C_R[s][i]$ ($C_R'[s][i]$) is the number of right (left) completions of $t_{i,s}$ unmatched by any string in R.
The recurrence relationship for computing C_R similar to that of C_S:

$$C_R[s][l-r+1] = \begin{cases} 0, \text{ if } t_{l-r+1,s} \text{ is matched in S} \\ 1, \text{otherwise.} \end{cases}$$

$$C_R[s][i] = \begin{cases} 0, \text{ if } t_{i,s} \text{ is matched in S} \\ \sum_{x \in A} C_S[\hat{s}.x][i+1], \text{otherwise.} \end{cases}$$

C_R' can be computed using a similar recurrence relation as for C_R.
$D_S[s][i] = C_S[s][i] \wedge C_S'[s][i]$ is a boolean value which specifies whether the template $t_{i,s}$ can be used for the creation of detectors.
$D_R[s][i] = C_R[s][i] \times C_R'[s][i]$ gives the number of strings in the current detector set that are unmatched by $t_{i,s}$. $D_R[s][i]$ is 'valid' if $D_S[s][i]$ is true..

To generate detectors, we pick the template corresponding to the maximum valid element of D_R. The remaining blank positions of the template are filled up by traversing the D_R array to the left and to the right, and adding at each step a symbol from A depending on which one represents the valid template with the highest number of strings not yet matched by the current detector set.
The C_R and C_R' arrays need to be updated to account for the new detector added. This is done by zeroing those elements of the arrays that correspond to templates present in the detector and re-computing the elements which are affected by this change.
The process is continued until all valid elements of D_R are equal to zero (which would indicate that the entire coverable non-self space has been covered by the detectors) or a pre-specified N_R number of detectors are generated.
This extended algorithm (*m*-ary GDGA) has the advantage of not requiring the data to be represented in binary form thereby retaining the information content of the string.

Space Complexity

The dimension of the C and D arrays is $m^r \cdot (l-r+1)$ While the C arrays are binary and hence require a bit for each element, the D array elements require to represent a maximum value of m^{l-r}, which would require $(l-r) \cdot \log_2 m$ bits. Hence the space complexity of D arrays is $O(m^r \cdot (l-r)^2)$ and that of C arrays is $O(m^r \cdot (l-r)^2)$.

Time Complexity

The time required for generating a detector is the sum of (a) $O(m^r \cdot (l-r))$ to find the template corresponding to the maximum valid element of D_R, (b) $O(m \cdot (l-r))$ to fill up the $(l-r)$ unspecified positions of the template, since there are position requires considering the *m* possibilities and (c) $O(m^r \cdot (l-r))$ for updating the C arrays. The time complexity of the algorithm is hence $O(m^r \cdot (l-r) \cdot N_R)$.

3 Results

The *m*-ary GDGA has been tested on three types of data – time-series, assembler instructions and system call traces. The behavioral patterns of each of the systems are expressed in strings of a finite alphabet by an appropriate mapping. The results are compared with those of binary GDGA.

3.1 Time Series Data

The application of negative selection algorithm to anomaly detection in time series data has been presented by Dasgupta and Forrest (1996). First, time series data is collected that adequately expresses the behaviour of the system. The range of variation [MIN, MAX] of the data is determined and the values are encoded in binary using a pre-specified number of bits. Then a window of a size appropriate to capture the semantics in data pattern is slid along the binary encoded time series and the binary string

outlined by this window is stored as a self string. The detectors are generated based on the resulting self strings using the negative selection algorithm. In the present scheme, we modify the encoding method by using an *m*-ary alphabet instead of a binary one. For example, with the rest of the procedure remaining the same, we may choose to map each point in the time series into a set of sixteen symbols corresponding to sixteen equally spaced points in the range [MIN, MAX], thereby obtaining self strings of a hexadecimal alphabet. Once the self strings have been generated by this process, *m*-ary GDGA is used to generate detectors.

The time series we consider the Mackey Glass series. The Mackey-Glass equation is a time delay differential equation that has been proposed as a model of white blood cell production. It is given by

$$\frac{dx}{dt} = \frac{ax(t-\tau)}{1+x^c(t-\tau)} - bx(t) \qquad (1)$$

The values of the constants a, b and c are generally taken as 0.2, 0.1 and 10 respectively. τ is the delay parameter. Normal data is computed using $\tau = 30$ and anomaly is introduced using $\tau = 17$. Fig 1(a) shows the Mackey Glass series with anomaly introduced in the region [500, 1000].

The self strings are generated for the normal data using the encoding scheme discussed above and detectors are generated for the same using *m*-ary GDGA. The strings corresponding to anomalous data are then matched with the detectors. A match would indicate the presence of anomaly. The success of the algorithm in detecting anomaly is measured in terms of the percentage of anomalous strings that are detected.

Table 1 illustrates the results for binary GDGA and *m*-ary GDGA. Values for 1500 points of the Mackey Glass series were computed using fourth order Runge-Kutta method (Wan). Anomaly was introduced in the region [500, 1000]. The data was encoded using five bits. Hence the points were mapped to a set of thirty-two distinct values.

The self strings used to generate detectors for binary GDGA were a concatenation of four five-bit binary encoded data points. The *m*-ary GDGA used self strings composed of four data points encoded in an alphabet of 32 symbols.

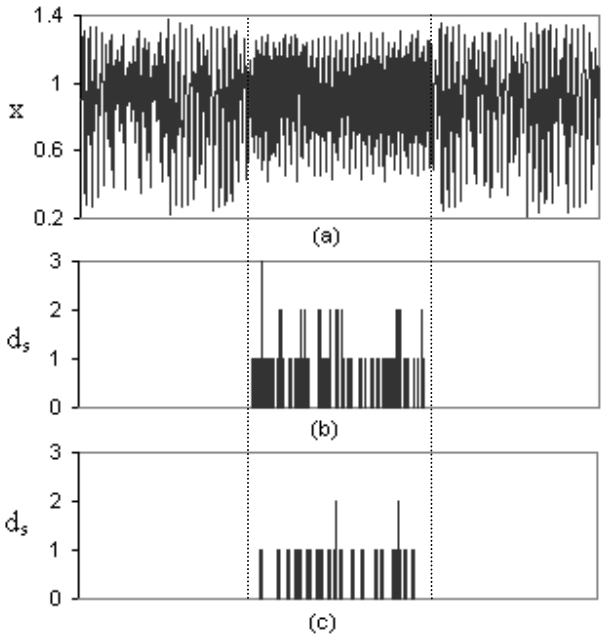
Each symbol of the 32-ary alphabet corresponds to 5 symbols of the binary alphabet. Hence the matching length r = 10 for m = 2 can be compared with r = 2 for m = 32. Comparing the success for $N_R = 100$, we see that there is a large difference in the success percentages (46% and 19% respectively for binary GDGA and *m*-ary GDGA). This can be explained as follows. The number of strings matched by each detector is given by (Wierzchoń 2000)[2]:

$$D_m(l,r) = m^{l-r-1} \cdot [(l-r)\cdot(m-1)+m] \qquad (2)$$

[2] This formula is valid only for $r \geq l/2$. A general method to calculate this value is presented by Esponda and Forrest (2002).

$D_2(20, 10) = 6144$ and $D_{32}(4,2) = 3008$. Thus the number of strings matched by a binary GDGA detector is more than twice that matched by an *m*-ary GDGA one. This results in a greater coverage by the former and thereby a higher success rate. However, in *m*-ary GDGA, we retain the semantics of the data unlike in binary GDGA where the r-contiguous match takes place across boundaries (the position between two 5-bit encoded data points defines a boundary.)

Figures 1(b) and 1(c) show the performance of binary and *m*-ary GDGA respectively with $N_R = 100$. d_S is the number of detectors which matched with a self string. The height of the vertical lines in the graphs corresponds to the number of detectors activated (d_S) when anomalous patterns were found. The dotted lines across the graphs indicate the region of anomaly. The blank regions of the graphs ($d_S = 0$) indicate no detection of anomaly. This may be either because anomaly does not exist in the strings corresponding to those regions of the time series or because the given set of detectors is incapable of detecting the anomalous strings.



(a) Mackey Glass series for 1500 points. Anomaly is introduced in the region [500, 1000] (indicated by the dotted lines). (b) and (c) show the number of detectors, given by height of vertical lines, that were activated when anomalous patterns were found. The size of the detector set (N_R) used = 100 and the self set size (N_S) = 749. (b) is the performance of binary GDGA and (c) is that of *m*-ary GDGA.

Figure 1: Performance of Binary and *m*-ary GDGA for Detection of Anomaly in Mackey Glass Series

Table 1: Anomaly Detection in Mackey-Glass Series

N_R	Success	
	Binary GDGA m = 2, l = 20, r =10	***m*-ary GDGA** m = 32, l = 4, r = 2
30	9%	6%
50	17%	7%
100	46%	19%
200	70%	37%
Encoding parameters		
Win. size = 4 Win. shift = 2 Self size N_S = 749		

Success is the percentage of anomalous strings that are detected. N_R is the number of detectors being used. Alphabet size (m) is the number of symbols in the alphabet being used for encoding. Self length (l) is the number of symbols that the self string is made of. Matching length (r) is the matching threshold for the partial match. Win. size is the number of data points that are concatenated (after being encoded) to create the strings. Win. shift is the number of data points that are slid across to get the next string. N_S is the number of self strings that are created.

It is seen that binary GDGA performs better than *m*-ary GDGA in the case of anomaly detection in time-series data such as the Mackey Glass time series. The detectors generated by the former algorithm cover more string space than that of the latter. This contributes to the better performance of binary GDGA. However, *m*-ary GDGA has the advantage of generating detectors for strings which encode the system characteristics in an alphabet of greater size. This would be relevant when there is a requirement to encode data in a higher alphabet.

3.2 Assembler Instruction Data

When a virus attacks a binary file it results in the modification of the file, which may be viewed as an anomaly in the system. There are various ways in which viruses modify binary files. Here we consider two methods. The first is a simple case of Single Mutation wherein a single instruction in the file is modified. The second is a more complicated modification in which virus code is attached to the file, resulting in the infected file becoming a source of infection. Most viruses which affect executables are of the second form.

To detect anomaly in binary files, the file to be protected can be represented in the form of a string of assembler instructions. The size of the alphabet is the size of the instruction set. Self strings can be constructed in by sliding a window across the assembler instructions in the same manner as discussed in Section 3. Detectors are generated for the resultant self strings which are used for anomaly detection in the file.

The detection of anomaly in binary files modified by single mutation and by a file infector virus are discussed below.

3.2.1 Single Mutation

Tests were conducted for a C file compiled for a Pentium processor[3]. `gcc` was used to generate the assembler code from the C code using the `-S` option. The assembler instructions were extracted and strings were generated by sliding a window across the trace. Detectors were generated for these strings. The assembly code was then altered by changing a single instruction, and the strings thereby generated were checked for anomaly. The *m*-ary GDGA was used to generate detectors.

The success of these detectors in identifying anomaly is shown in Figure 2. The most commonly used instructions were taken as the alphabet, with size of 16.The results were averaged over 1000 iterations. Each set had 3 detectors. Considering multiple detector sets is relevant in the case of a distributed system. There may be many nodes in the system and each could be given a set of detectors. The overall probability of anomaly detection increases with greater number of detector sets.

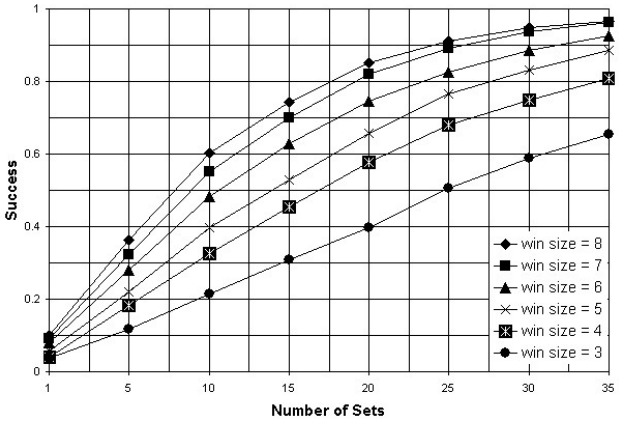


Figure 2: Performance of *m*-ary GDGA on Assembly Instruction Data

Alphabet size m = 16, win. shift = 2, matching length r = 2, number of self strings N_S = 35, number of detectors per set N_R = 3. Number of iterations = 1000.

The effect of varying the window size, and thereby the string length, is demonstrated in Fig 2. Increasing the string length while keeping the window slide value and matching length constant results in an increased success rate. This is because a detector of greater length covers a greater area of non-self, the matching length remaining

[3] The method of creation of strings is given in the appendix.

the same. However, the number of holes, or non-coverable non-self space (D'haeseleer 1996) increases as well.

3.2.2 File Infector

File infector viruses attach themselves to executable code. They activate when an infected file is run, and then usually remain in memory. Thereon any non-infected executable that runs becomes infected.

A test virus was generated using a commonly available virus creation tool known as Instant Virus Production Kit (IVP). The virus was programmed to infect mem.exe, a DOS executable. The description of this process is given in the appendix.

The self set was created by disassembling the executable and sliding a window of size four across the trace of assembler instructions, moving in steps of two, as in Section 3.3.2. Detectors were created for the self strings. The executable was then infected using the test virus, and the strings generated from the infected file were matched with the detectors to check for the presence of an anomalous sequence.

Table 2 shows the performance of binary and *m*-ary GDGA. The process of testing was as follows. First, a large pool of detectors (1000) was created for the 4854 self strings corresponding to the uninfected mem.exe. Then a given number of detectors were randomly chosen from the pool and were used to detect anomaly in the set of strings corresponding to the infected file. The results are averaged over 1000 iterations.

Table 2: Anomaly Detection in Assembler Instruction Data

N_R	N_D	
	binary GDGA	***m*-ary GDGA**
10	0	0
50	1	2
100	3	4
200	6	7
500	14	18
Parameters		
$N_S = 4854$	Win. Size = 4	Win. Shift = 2

N_R is the number of detectors. N_D is the number of anomalous strings that were detected by the N_R detectors. N_S is the number of self strings. The number of anomalous strings was 151. The results are averaged over 1000 iterations.

Although the detectors do not detect the total number of anomalous strings (=151), the presence of at least one anomalous string would indicate a modification in the file. For self sets which vary over time, a threshold value for the number of allowable anomalous strings could be set. If the number of anomalous strings detected exceeds this threshold value then the system may be deemed to contain anomaly.

The table demonstrates the relative performances of binary and *m*-ary GDGA. The performance of *m*-ary GDGA is marginally better than that of binary GDGA. This indicates the advantage *m*-ary GDGA has when the information is encoded in a higher alphabet.

The size of the alphabet (=147) considered here is much larger than that considered in Section 3.3.1. This results in a lesser fraction of area being covered by these detectors compared to the detectors of smaller alphabet. To illustrate this, we calculate the area covered by the detectors generated in Sections 3.3.1 and 3.3.2, using Equation (2), and divide it by the area of the total string space. In both cases, four instructions are concatenated to make a self string, and the window slide value is two. The matching length, measured in number of instruction, in both cases is two. The alphabet size, *m*, of detectors R_1 in Section 3.3.1 is 16 and of detectors R_2 in Section 3.3.2 is 147. Using Equation (2), we have $D_{16}(4, 2) = 736$ and $D_{147}(4,2) = 64533$. The fraction of the total string space covered by R_1 is $736/16^4 \approx 1.12\text{e-}2$ and that by R_2 is $64533/147^4 \approx 1.38\text{e-}4$. Hence the accuracy of anomaly detection of R_2 is substantially lesser than R_1.

3.3 System Call Data

The efficacy of short sequences of system calls as discriminators for several types of intrusion has been discussed by Forrest *et al.* (1996) and D'haeseleer *et al.* (1996). Here we present results of using binary and *m*-ary GDGA to detect anomaly in a system call trace of a Trojan code[4] for `login` (UNM).

3.3.1 Intrusion detection based on login system call trace

`Login` is a program used when signing into a system. It is also used to switch from one user to another. Here, the trace of system calls issued by `login` are used to classify system behaviour. An anomalous behaviour would be indicative of a possible intrusion in the system.

A normal `login` trace (LoginNormal) was used to collect the behavioral patterns of legitimate usage of the program. The data contain pairs of values in the form (pid[5], system call). Strings were created by sliding a window of size four across system calls with the same pid, with windows shift value as two. Detectors were generated for these strings using *m*-ary GDGA. Then similar

[4] A Trojan code (Trojan Horse) has been defined by Dan Edwards as "a malicious, security-breaking program that is disguised as something benign." (Foldoc).

[5] Since there may be multiple processes created by a single program, it is required to indicate the process ID (pid) as well.

strings were constructed for a Trojan version of `login` (LoginRecovered) and were checked for anomaly using the detectors previously generated. The results are presented in Figure 3(a).

The data considered in Fig. 3 is the trace of system calls with the pid of 801 from LoginRecovered. The trace contained 386 system calls, from which 192 unique strings were constructed. The x-axis measures the position of the anomalous string in units of system calls. The height of the vertical lines corresponds to the number of detectors that matched with anomalous strings. Hence the lines in the region [0,14] indicate that anomalous behaviour is detected in the first fourteen system calls

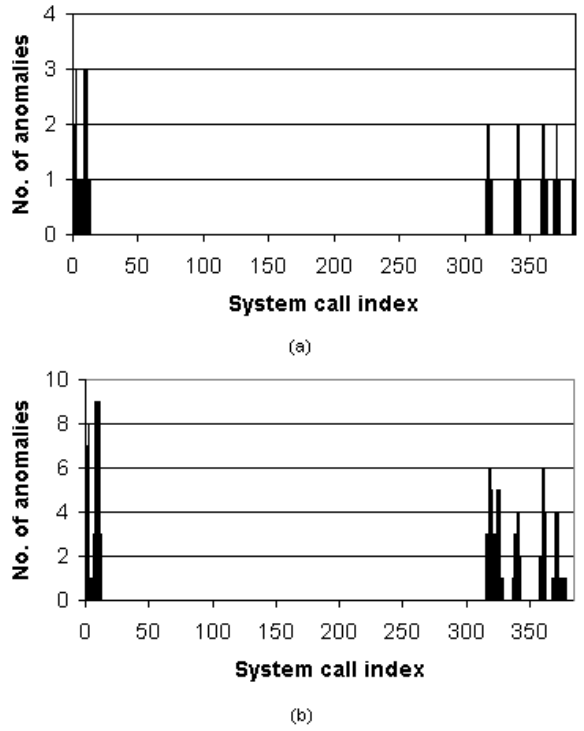


Figure 3: Performance of *m*-ary GDGA on System Call Trace of Trojan Version of Login.

(a) anomaly detection by *m*-ary GDGA detectors. Alphabet size m = 164, win. size = 4, win. shift = 2, string length l = 4, matching length r = 2 (b) anomaly detection by binary GDGA detectors. Number of bits used to encode a system call nb= 8, win. size = 4, win. shift = 2, string length = win. size x nb = 4 x 8 = 32, matching length r = 8.

Number of self strings from which detectors were generated, N_S =178. Number of detectors N_R = 89. Number of strings constructed from the anomalous trace = 192.

Figure 3(b) demonstrates the performance of binary GDGA on the same data. The 164-symbol alphabet was encoded in 8 bits. The binary encodings of four contiguous system calls were concatenated to form thirty-two bit strings. It is seen that more detectors are matched in 3(b) compared to 3(a). The explanation for this has been given in Section 3.1.

3.3.2 False positives

We consider here the occurrence of false positives in the detection of anomaly in system call data using binary and *m*-ary detectors.

LoginNormal consists of seven relevant traces. An exhaustive detector set is generated for each of these traces. The detector set corresponding to a trace is then matched against the strings corresponding to the entire LoginNormal trace. The number of system call sequences tagged as anomalous by these detectors is noted. Since the each of the traces that are checked for anomaly are normal data, the detection of anomaly in them would be a false positive. Table 3 shows the occurrence of false positives in anomaly detection using binary and m-ary GDGA detectors.

Table 3: False Positives in Anomaly Detection in System Call Data Using Binary and *m*-ary GDGA

Trace PID	binary GDGA		*m*-ary GDGA	
	F.P.	Exh. N_R	F.P.	Exh. N_R
19563	220	242	74	130
2188	109	238	74	130
4938	109	238	74	130
509	109	238	74	130
598	109	238	74	130
8954	192	232	72	129
9280	109	238	74	130

Trace PID is the process ID of the trace based on which the detectors have been generated. F.P. is the number of false positives, or number of (normal) system call traces which were flagged as anomalous. For each case, the detectors were matched against the entire LoginNormal trace to find the number of false positives. Exh. N_R is the size of the exhaustive detector set.

Win. size = 4. Win. shift = 2. Alphabet size m = 164. Size of LoginNormal trace = 5937.

An exhaustive detector set is considered so as to present a definitive measure of false positives. As is seen from the table, *m*-ary GDGA detectors have a lower count of false positives that binary GDGA detectors. This is due to the retention of semantics of the data by *m*-ary GDGA.

4 Discussion

4.1 Situation favouring *m*-ary GDGA

The utility of *m*-ary GDGA in detecting anomaly in various forms of data has been presented in Section 3. *m*-ary GDGA would be preferred when a binary encoding would result in a loss of information content of the data.

In some cases, even when data originally encoded in a higher alphabet is considered, binary GDGA may perform better than m-ary GDGA[6]. However, as is explained below, this occurs when the difference between alphabet size *m* and the next highest power of two is small. For higher values of this difference, *m*-ary GDGA would perform better.

It was seen is Section 3.1 (time-series data) that binary GDGA had a better performance than m-ary GDGA. The alphabet size considered was thirty-two. Hence five bits were used to encode a symbol, which were then concatenated to create binary strings of length twenty. In general, if the alphabet size is m and the m-ary string length is *l*, then the binary string length l_B would be given by $l_B = 2^{\lceil \log_2 m \rceil} l$. When *m* is a power of two then each of the 2^{l_B} binary strings would be valid (self or non-self) strings. However, if *m* is not a power of two then there are $2^{l_B} - m^l$ strings which are invalid, i.e., they do not correspond to any point in the unencoded space. However, binary GDGA cannot distinguish between an invalid string and a non-self string. Hence detectors are generated to cover the invalid string space as well. When $2^{l_B} - m^l$ is small then this detector 'wastage' is small. However, for larger values, this may present a problem. In other words, when the value of *m* is significantly lesser than the next higher power of two, then the number of invalid binary strings is higher, resulting in a lower performance of binary GDGA compared to m-ary GDGA (which does not encounter this problem). This behaviour is seen in Section 3.3 (assembler instruction data), where *m* = 147, which results in the size of invalid binary string space to be nearly eight times the valid space.

To counter the problem of wasted detectors, the invalid string space may be included into the self space so that detectors are not created to cover that space. However, this once again would pose a problem, since the size of invalid space may be many times the valid space, as was seen in the previous example.

m-ary GDGA allows retaining the original encoding of data. Hence the problem of invalid strings is not encountered. The utility of *m*-ary GDGA would be higher in these circumstances.

[6] To create self strings from this m-ary data for binary GDGA to operate upon, each symbol of the m-ary alphabet can be encoded into a binary string of length $\log_2 m$.

4.2 Choosing the matching length

For a given self set of size N_S, string length *l* and detector set size N_R, the matching length *r* can be chosen experimentally so as to maximize accuracy of detection.

For a given matching length *r*, detectors of a given number are generated and are matched against the entire non-self set. The number of undetected non-self strings $N_{N'}$ is noted. The optimum value of *r* would be that corresponding to least value of $N_{N'}$.

A smaller value of *r* would enable a detector to cover more non-self space (as can be seen from Equation 2) and hence would be expected to result in a lower value of $N_{N'}$. However at the same time, the number of holes, N_H, in the space also increases, thereby increasing the number of strings that are undetectable by the given set of detectors. Hence the optimum value of *r* can be found by noting the point at which the value of $N_{N'}$ reaches the minima.

The disadvantage of this method is that it is computationally expensive. However when the non-self set is not very large, this method can be used.

5 Conclusions

The negative selection algorithm has the advantage of distributability. Any subset of the complete detector set can be used for anomaly detection. This is useful when there are multiple anomaly detection nodes.

The performance of the *m*-ary GDGA has been studied. It has the advantage of retaining the semantics of the data. This is particularly relevant to system call data and assembler instruction data where a binary encoding of the data would result in a loss of the semantics of information. Its disadvantage is its greater time and space complexity compared to binary GDGA.

The lesser number of false positives in the case of *m*-ary GDGA compared to that of binary GDGA indicates the better performance of the former when the original encoding is non-binary.

The usage of *m*-ary GDGA is particularly advantageous when the value of alphabet size *m* is significantly lesser than the next higher power of two.

6 Future Work

Current work is in the direction of evaluating the utility of negative selection to various data (Kim and Bentley 2001). An equivalent 'positive selection' method is being used for this evaluation, wherein the entire set of self strings is used for anomaly detection based on the r-contiguous matching rule.

An understanding of the r-contiguous distance and its related space would be important in creating an improved detector generating algorithm. The possibility of embedding the r-contiguous space into Euclidean space (Bourgain 1985) is presently being studied for the purpose of

visualizing the former space, and for borrowing ideas from coverage algorithms developed for Euclidean space.

Acknowledgments

The present work has been performed as part of an undergraduate project work at the DaimlerChrysler Research Centre India Pvt. Ltd., Bangalore. The author wishes to thank Akash Narayana, Dr. R. Raveendran and Dr. Shanmukh Katragadda for their valuable guidance, supervision and encouragement. The author is grateful to Dr. Roland Haas for having offered this opportunity and for the encouragement and support during this work.

References

J. Bourgain (1985) On Lipshitz Embedding of Finite Metric Spaces in Hilbert Space. In *Israel J. Math. 52*, pp. 46-52.

S. Forrest, A. S. Perelson, L. Allen and R. Cherukuri (1994) Self-nonself discrimination in a computer. In *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, 202-212. Los Alamitos, CA: IEEE Computer Society Press.

S. Forrest, S. A. Hofmeyr and A. Somayaji (1996). A sense of self for unix processes. In *Proceedings of the 1996 IEEE Symposium on Research in Security and Privac*y, Los Alamitos, CA: IEEE Computer Society Press.

D. Dasgupta and S. Forrest (1996). Novelty detection in time series data using ideas from immunology. In *ISCA 5th International Conference on Intelligence Systems*, Reno, Nevada

P. D'haeseleer, S. Forrest and P. Helman (1996). An Immunological Approach to Change Detection: Algorithms, Analysis and Implications. In *Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy*, IEEE Press.

S. A. Hofmeyr, S. Forrest, and A. Somayaji (1998). Intrusion detection using sequences of system calls. *Journal of Computer Security*, 6:151-180

S. T. Wierzchoń (2000). Discriminative power of the receptors activated by *k*-contiguous bits rule. *Journal of Computer Science and Technology. Special Issue on Research Computer Science*, vol. 1, no. 3, 1-13.

J. Kim and P. J. Bentley (2001). Evaluating Negative Selection in an Artificial Immune System for Network Intrusion Detection. *Genetic and Evolutionary Computation Conference 2001* (GECCO-2001), San Francisco, pp.1330 – 1337.

F. Esponda and S. Forrest (2002). Detector Coverage under the *r*-contiguous bits matching rule. *University of New Mexico Technical Report TR-CS-2002-03*.

Websites:

UNM, Computer Immune Systems, Data Sets, http://www.cs.unm.edu/~immsec/data/login-ps.html

LoginNomal,http://www.cs.unm.edu/~immsec/data/login-live-unm.tar.gz

LoginRecovered, http://www.cs.unm.edu/~immsec/data/login-recovered.int.gz

E.Wan, Time Series Data, http://www.ece.ogi.edu/~ericwan/data.html

HofmeyrStide:http://www.cs.unm.edu/~immsec/software/

Foldoc, http://foldoc.doc.ic.ac.uk/foldoc/.

Appendix

(a) Creation of *m*-ary strings from C code.

m-ary strings are created from a given C code (sfile.c) using the following piped sequence of commands:

```
gcc -S - sfile.c -o tmp/sfile.s  ;
cat tmp/sfile.s   |
sed -n 6~1p       |
cut -f2           |
cut -d" " -f1     |
grep -v "[:|\.]"  > tmp/sfile.log
```

The file sfile.log now contains the sequence of assembly instructions corresponding to sfile.c. This file is then converted to a sequence of *m*-ary strings by a replacing each instruction with a unique identifier, and then sliding a window across this data.

(b) Virus creation using Instant Virus Production Kit

Instant Virus Production Kit (IVP) is a virus creation tool that can be used to create viruses by specifying certain parameters. A configuration file is used which has parameters such as given below:

- Infection type. exe, com, both, trojan.
- Overwriting or appending upon infection.
- ID the virus uses for checking for infection.
- Encryption.
- Change Directories.
- Set the INT24 handler.
- Max number of files to infect at runtime.
- Time based activation options.

IVP generates a self-propagating virus assembly code based on these parameters. The code is compiled, linked and then executed to infect a dummy executable file. This dummy file becomes the source of infection

This tool was used to create a virus which was customized to infect mem.exe by specifying the name of this file in the search string of the virus.

Self-Assertion versus Self-Recognition: A Tribute to Francisco Varela

Hugues Bersini
IRIDIA-CP 194/6
Université Libre de Bruxelles
50, av. Franklin Roosevelt
1050 Bruxelles – Belgique
bersini@ulb.ac.be

Abstract

Ten years ago, a group of researchers, led by Francisco Varela, were proposing an alternative vision of the immune system main behavior and function. I was part of this group. This new vision saw the immune system not as behaving distinctively with self and non-self or according to any dichotomy imposed a priori and from outside (the self-recognition vision), but rather as behaving in a unique way. From this indifferent behavior, any external impact would progressively been treated in two different ways, reactive and tolerant, but now, consequently and from inside the system (the self-assertion view). This paper will recall, through a very artificial simulation, the difference existing between these two visions. Also at that time, we believed that, from an engineering perspective, this new vision, emphasizing more the adaptability and the need for endogenous constraints than the recognition and the defensive ability, although less obvious to accept than the classical defensive one, should be more beneficial. These last ten years proved that we haven't been convincing enough, and in this paper I resume the crusade.

1 INTRODUCTION

Ten years ago, Varela, Coutinho and Stewart (Varela et al., 1988; Varela and Coutinho, 1991; Stewart, 1994) were proposing and defending a new vision of the immune system, largely in the continuation of Jerne's intuition and model (Jerne, 1974), in which the "self" and "foreignness" dichotomy collapses, for the system is complete unto itself. Based on simulations of the immune idiotypic network and some experimental data, they published several papers, although not in the mainstream journals of immunology. In an idiotypic network, there is no intrinsic difference between an antigen and an antibody, and any node of the network can bind and be bound by any others. My role is this group was two-fold. At that time not a biologist and still not today, I was responsible with Vera Calenbuhr and Vincent Detours for the development of a series of computer simulations that have been described in (Detours et al., 1994; Bersini and Calenbuhr, 1996; Calenbuhr et al., 1996). I was also responsible for trying to initiate the influence of this new "reading" of this biological system for the conception of engineering artifacts. A mission I tried to fulfill in (Bersini and Varela, 1993; Bersini, 1999).

Although we pushed hard for this alternative vision, we need to admit today that the classical view of the immune system as a defensive system, first able to distinguish between dangerous and innocent external impact, and thus to defense against the dangerous ones, has been the most influential one from an engineering perspective. It was clearly the most appealing to adopt, but it's a pity. First, this is a vision that is facing more and more opposition among the biologists themselves. But beyond that, I am convinced that we don't need to know how the immune system distinguishes, if it does so, between good and bad stimuli, in order to build performing two-classes classification system or any pattern recognition mechanism. Also, we don't need to know how the system creates good markers of self, if it does so, to build performing clustering and self-organizing systems. And finally, we don't need to know how the system protects the body from external damages, if it does so, to build good protective system for computers. The Panda antiviral system that by computer has adopted for one year now is one good illustration, among many, of that. I don't believe the developers of such effective software needed to know anything about how the immune system fights natural virus to develop their system for artificial ones.

In the first section, largely relying on a recent excellent survey of the Stanford Encyclopedia of Philosophy (Tauber, 2002), I will try to summarize what main lines of criticisms attack the vision of immune system as able to distinguish self from non self and able to protect from non-self. In the second section, I will present a very simple software simulation that will make easier to understand the difference between the self-recognition and the self-assertion views. This simulation is very reminiscent of a lot of simulations that we published years ago, although I'll try to simplify it to the basics in order to really shed the light on the key differences.

Finally, the last section will try to defend why the self-assertion view should inspire in a more creative way the conception of engineering artifacts. This vision leads to strongly adaptive systems, both parametrically and

structurally, but whose adaptability mainly aims at satisfying endogenous constraints instead of responding to exogenous impacts. This constraint satisfaction might provide the system with several adaptive advantages as, for instance, the capacity to respond to a large diversity of external stimuli and to memorize in an economical way a repertoire of adapted responses when facing a non-stationary environment.

2 THE PROBLEMS WITH SELF AND NON-SELF

Although it's important not to confuse the alternative view proposed by Polly Matzinger (Matzinger, 2002), today best known critics of classical immunology, with the one proposed by Varela's group, part of the criticisms addressed by Matzinger to classical immunology has to be taken as important flaws of this classical view. Why the immune defenses do not protect us from the air we breathe, the food we eat, the fetuses we carry, the tumors that kill us (even then it should)? Why are a lot of our lymphocytes autoreactive without any sign of autoimmune diseases? Indeed, a lot of evidences in recent years have shown that autoimmunity is a normal finding in healthy individuals. Clearly the problem with self and non-self lies in the determination, namely the nature and the location, of the frontier. The designation of "self" and the "other" ignores that such neat divisions were adopted with a certainty that remain problematic.

One first relaxation to the self/non-self dual view of the immune system is to maintain the duality, i.e. the immune system keeps two ways of being in response to external impact: defensive and tolerant, but not depending on an evasive frontier to cross. It is the position adopted by Matzinger who insists in getting rid of the self/non-self discrimination as the central tenet of immunology. What she proposes instead is an immune system that just fights what is dangerous for it. So the dichotomy is maintained but self/non-self is simply replaced by dangerous/inoffensive. The fact that this move finally consists in this simple semantic substitution makes a lot of immunologists very skeptic against Matzinger position. However it appears that fighting danger rather than foreignness entails doctors to adopt therapeutic strategies that show great successes for certain serious diseases.

Now exploring more logically Matzinger's position, and although the full model is still somewhat confused, it is important to understand better what does the immune system see as dangerous and why it does so. One view, the less radical one, would see the danger as resulting from some specific characteristic of the external perturbation. It might be an additional feature of the invading antigen. In such a case, from the outside, the external impact will be, prior to any interaction, dangerous or not, and the immune system would still need to somewhat behave in a dichotomous way, first recognizing the danger then fighting it. The external environment of the immune system will still be separated in two zones: a dangerous and an inoffensive one. This interpretation of what is dangerous or not is not such an exciting one, because it still demands from the system the ability to discriminate and to defend. The self/non-self frontier is just re-defined but still exists outside the system.

The most radical view, and for reasons to be discussed later, makes Matzinger and Varela closer than they appear to be (in her "science" article Matzinger said that after many years of finding Varela's model intriguing she finally agrees). Varela's view would see the danger as a consequence of the interaction between the external impact and the current state of the immune system. In such a case, a stimulus is no more dangerous per se, but is dangerous in the current context of the immune system. An outside separation in two classes, making the immune system behaves in two ways, simply collapses. We remain with an immune system behaving in one only way but, depending on its current state and the nature of the impact, proposing different responses to it. For instance, a same external impact could drive the system to react differently at different times.

The reason why this second, more innovative interpretation, is akin to Varela and his group vision can be easily understood by reading the following excerpt from the Stanford Encyclopedia (Tauber, 2002) about the later vision:

"*When the immune system is regarded as essentially self-reactive and interconnected, the meaning of immunogenicity, that is reactivity, must be sought in some larger framework. Antigenicity then is only a question of degree, where "self" evokes one kind of response, and the "foreign" another based not on it intrinsic foreignness, but rather because the immune system sees that foreign antigen in the context of invasion or degeneracy. …. In the Jernian network, "foreign" is defined as perturbation of the system above a certain threshold. Only as observers do we designate "self" and "non-self". From the immune system perspective it only knows itself…. While host defense is a critical function, it is hardly the only one of interest. Indeed the immune system might be regarded as primarily fulfilling an altogether different role if its phylogeny is carefully examined…. Immune reactivity is determined by context where agent and object played upon each other….."*

3 A VERY ARTIFICIAL MODEL TO DISTINGUISH THE TWO VISIONS

In this section, I will describe a very simple model built in a two dimensional space and very reminiscent of several models that I did build years ago with my colleagues John Stewart, Vera Calenbuhr and Vincent Detours (Detours et al. 1994, Calenbuhr et al., 1996, Bersini and Calenbuhr, 1996). It will provide an easy to understand illustration of the difference between the self-recognition and the self-assertion visions.

We will suppose that any immune cell (they could be antibodies) be identified by its position in a two

dimensional space. In agreement with the key-lock binding of immune cells with antigens, we will also suppose, like indicated in figure 1, that any immune cell exerts an affinity in a zone symmetrically situated with respect to its position. What we want to model by this artificial construction is the possibility for a cell to bind an antigen when it presents a shape symmetrical with respect to the one of the antibody. The affinity is not restricted to the symmetrical position but extends to a square domain of size L, the strength of the binding decreasing with the distance to the center of the square.

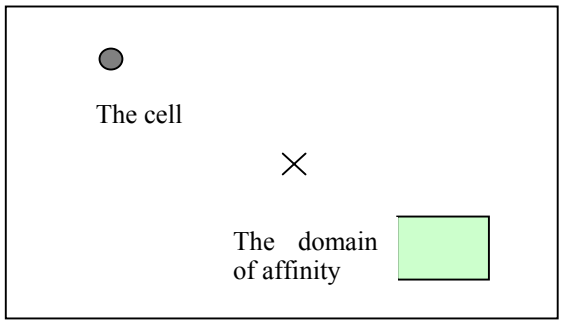


Figure 1: One cell and its symmetrical domain of affinity

At every simulation step, a new cell is randomly recruited anywhere in the system. It is added with an initial concentration Ci(0), C for concentration and i indexing the cell. Suppose the center of symmetry Xo, Yo. Suppose a cell i situated in position cx and cy and having concentration Ci(t) at time t . It will exerts an affinity on any cell situated in position x, y with value given by:

$$affinity = Ci(t)*(L - (|2Xo - cx - x| + |2Yo - cy - y|)/2)$$

So all cells will exert on any cell j a field of affinity *Affj* obtained by summing this affinity for all the cells currently present in the system:

$$Affj = \Sigma_i \, affinityOfCell_i$$

3.1 THE SELF-RECOGNITION VIEW

In the classical view, we will assume that only the antigens are subject to this field of affinity and, reciprocally, only the antigens exerts affinity on the cells.

At every time step, the concentration of the cell j evolves in the following way:

if (low < Affj < high) Cj(t) = Cj(t) +1

else

Cj(t) = Cj(t) – 1

if Cj(t) = 0 the cell j disappears from the system

In this case *Affj* is computed just by summing the field exerted by the antigens.

$$Affj = \Sigma_i \, affinityOfAntigen_i$$

Consitently with immunological facts, the cells will grow in concentration, i.e. simulating an immune response, if they receive a stimulating field in between two thresholds: *low* and *high*, whose precise values must be known for the simulation to run. The field must be sufficient enough but not too high due to the bell shape curve of the maturation and the proliferation of B lymphocites and antibodies.

From their side, the antigens will just decrease in concentration if they are bound enough by the immune cells. Take an antigen j, if finding enough cells to bind it i.e. *if Affj > low*, it will decrease in concentration according to:

if (Affj > low) Cj(t) = Cj(t) – k(Affj/low)*

k is a time rate

if Cj(t) = 0 the antigen j disappears from the system

The simulation proceeds as follows. Initially, cells are recruited randomly in the system, but in the absence of antigens, so with no stimulating field, they can't survive and disappear as soon as they get in. When an antigen enters the space, the simulation behaves as illustrated in figure 2.

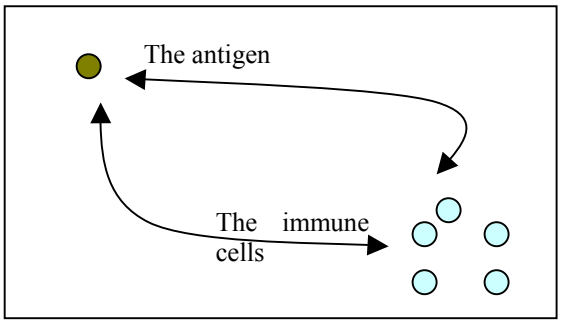


Figure 2: The reciprocal stimulation and elimination of antigen and immune cells

The antigen, now by the field of affinity it exerts symmetrically to its position, allows some cells to survive and to grow in concentration. These cells in turn exert a suppressing field on the antigen. The antigen will decrease in concentration until it disappears completely. Once it is cancelled from the system, the cells responsible for its disappearance are no longer stimulated and slowly

die, driving back the whole system to the initial situation: random recruitment of not surviving cells.

Playing with the concentration increasing and decreasing rates (for instance the constant *k*), the immune cells can take some time to disappear, akin to a sort of inertial memory of the antigen encounter. The next time a same antigen gets in, its cancellation will be faster like for any secondary immune response.

What needs to be understood, in contrast with the section to come, is that, in the classical case, cells show affinity only with antigen and not at all among themselves, although they occupy the same two-dimensional description space. Although nothing really differentiates an antigen from any cell, there must be a magical demon to tell the cells that the dot in the space is an antigen and not a cell.

3.2 THE SELF-ASSERTION VIEW

In this less classical view, all cells bind to all cells. To quote again the Encyclopedia: "*there is no essential difference between the "recognized" and the "recognizer", since any given antibody might serve either, or both, functions. Immune regulation is based on the reactivity of antibody with its own repertoire forming a set of self-reactive, self-reflective, self-defining immune activities*".

In the simulation now, the way we will compute the *Affj* received by any cell is as follows:

$$Affj = \alpha \Sigma_i \, affinityOfCell_i + \beta \, \Sigma_i \, affinityOfAntigen_i$$

This time, the affinity received by any cell is a weighted sum of the exogenous stimulation of the antigens and the endogenous stimulation of the cells themselves. Give a value 0 to α and you are back to the previous case. There is no way for any cell to discriminate between the exogenous and the endogenous impact. All impacts mix together to stimulate the change in concentration of any immune cell.

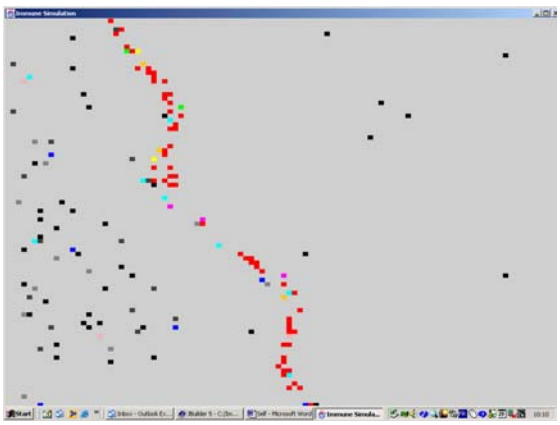


Figure 3: Snapshot of the self-assertion simulation

In the absence of any antigen, the simulation goes as shown in figure 3 (a snapshot of the simulation). The simulation slowly produces a sort of line or a band of self-sustained cells. Due to the way the affinity is computed (symmetrically with respect to the center of the space), cells in the line mutually stimulate themselves. A part of the line sustains another part of the same line. We speak of self-assertion since, indeed, this line can be roughly viewed as a signature of the immune self.

As shown in figure 4, first the system needs to be triggered off, and during the first time steps a lot of cells are recruited and very few are killed. During a second period, when the line of self-sustained cells begins to form, a lot of cells (not integrated in the line) are killed. This elimination phase can be roughly assimilated to the so called clonal selection phase taking place during the prenatal development and exercising a purging function of self-reactive cells. It is the period during which the tolerant zones are learned by the system itself. Finally the system tends to stabilize its rate of destruction and, while working at normal regime, integrates and kills new cells at a constant rate.

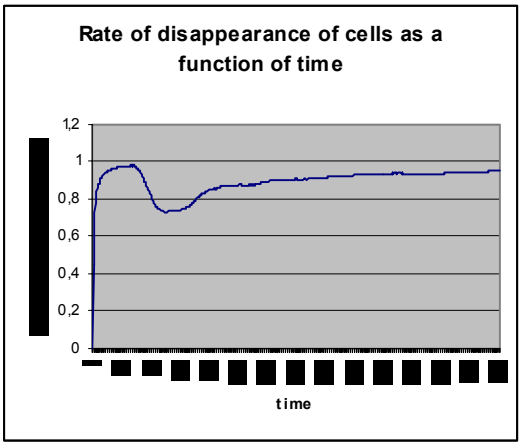


Figure 4: rate of disappearance of cells as a function of time

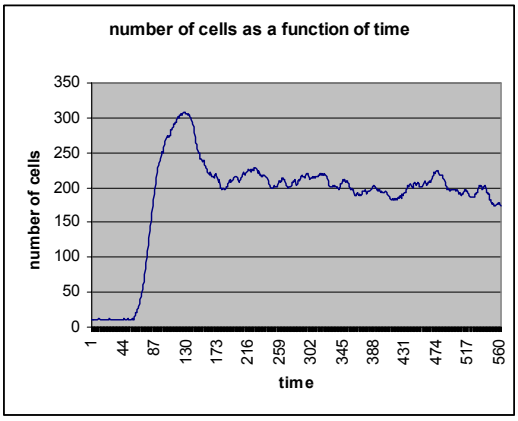


Figure 5. Number of cells as a function of time

In figure 5, plotting the number of cells as a function of time, again you can see the three successive phases of the simulation: first very few cells, then a short triggering period when a lot of new cells are recruited, and finally a stable regime.

One key observation is that the presence of the line divides the space in two zones: a reactive zone on the right and a tolerant zone on the left. If you add an antigen on the left, it will be tolerated since there is no cell on the right able to bind it. In contrast, an antigen on the right will be rapidly destroyed since a lot of cells on the left are still able to bind it. Basically the shape of the line is responsible for this division of the space in these two zones.

It must be clear that these two resulting zones are not shaped from the outside. There is no a priori division of the space into reactive and tolerant zones. This division is self-asserted by the system. The system creates, by its own evolution, its own zone of tolerance and own zone of reactivity. You might ask why a completely symmetrical simulation lead to unsymmetrical outcome. It is a simple artefactual effect of the random generation of cells that is amplified in time.

However the final separation of the space in a tolerant and a reactive zone will always be in relation with the history of the system. If you initially favor the recruitment in a given zone, this zone will naturally tend to become the tolerant one, a finding that qualitatively agrees with the Burnett's clonal selection theory.

This qualitative phenomenon i.e. the emergence of some geometrical patterns of self-sustained cells dividing the space in tolerant and reactive zones is very robust and largely independent of the values given to the parameters. This explains why I don't need to indicate the precise the values taken by the parameters of the simulation: α, β, *low, high*. The same qualitative outcome will be observed for a large range of values.

However, what's of crucial importance here is that no recognition and discrimination is at work. The system does not need to discriminate between an immune cell and an antigen, between self and non-self or along any prior arbitrary division applied to its biological environment.

4 TAKING AN ENGINEERING PERSPECTIVE

We are not biologists but are trying to be influenced by biology to create new ways of designing useful artifacts. As I already wrote in a previous paper (Bersini, 1999), I believe that the self-recognition interpretation of the immune system is not the most fruitful one. The basic reason is that this interpretation does not need biology to be expressed and understood. That the immune system can discriminate between two classes of external impacts can be easily translated into a classical pattern recognition problem. So far I haven't read any better ways of classifying, clustering data or constructing defensive systems, beyond classical ones, which have been discovered thanks to the immune analogy.

Also I don't want to pretend that the self-assertion view has been much more productive. Obviously, there have been fewer trials. In (Bersini, 1999) I discussed several engineering applications I was involved with that gained some benefits from applying here and there hints coming from immunology. The principal one that was implemented in all these application is the endogenous double plasticity inspired from immune networks. This endogenous double plasticity complies with the following principles:

1. the structural adjustments (akin to the recruitment of new cells) intermittently occur following a longer time scale than the parametric adjustment.
2. the structural plasticity amounts to the addition of new elements and the suppression of redundant elements from the system
3. again like in the artificial world shown above, the structural adjustments are dependent on the temporal evolution of the internal parameters (in the simulation, the current concentration of the cells). When and how to perform a structural change should depend on data related to the dynamics of the parametric change. So the network endogenous behavior and now exogenous criteria will guide these structural changes. Remember the immune system which only sees and knows itself.
4. these structural endogenous alterations have to be done in a network spirit by applying simple heuristics like "compensate for the weakest elements", "maintain diversity", "suppress redundancy".

In the same paper, I presented three practical illustrations of systems capable of evolving in time their structure and parameters while executing their task: neural net classifiers, autonomous agents that adapt by reinforcement learning, and controllers of chaotic systems.

In none of them, the biological influence was so strong to claim that I could not have done the same in the absence of any immunological knowledge, but in all of them, the way I tackled the problem, reinforcing the adaptability and the respect of the endogenous constraints, came from this knowledge.

5 CONCLUSIONS

The paper basic motivation is to better understand the difference existing between the classical self-recognition and the more "exotic" self-assertion visions of the immune system. Although the later is gaining more and more attention in the biological community, it is not

receiving the same attention in an engineering perspective. I believe it should.

We all need to admit that the immune algorithms, whatever they really turn out to be, did not provoke the same wave of interests as genetic algorithms or neural nets did for engineering applications. One key reason could be that in their initial presentation, both GA and neural nets were proposed in a very coherent and convincing way as simple algorithms, easy to implement, and associated with a precise and well-defined operational context: optimization for GA and pattern-recognition for neural nets. As a matter of fact, a lot of researchers discovered the whole problematic of optimization or pattern recognition by applying GA or neural nets.

Immune algorithms were never sold in such a persuasive way. No precise and complete algorithm was proposed and no clear operational context was associated with them: pattern recognition, defensive system, optimization, or robotics? Now, when maturing, researchers slowly realize that just playing with the initial basic GA or the initial neural nets does not give good results. What they do instead is to preserve some good mechanisms originating from this biological inspiration: population/selection based search or crossover for GA, multiplayer for neural nets, but turn them into a more operational form.

This is really what we are all doing today, based on our respective understanding of how the immune system behaves: gleaning here and there some inspirations and turning them into a more operational form. However, we should keep open our mind to more marginal voices, since if they are telling the truth, the radical revision they will entail in their community could have repercussions up to our own.

References

N. Jerne (1974) Towards a network theory of the immune system – *Annals of Institute Pasteur/Immunology (Paris)* – 125C: 373-389.

F. Varela, A. Coutinho, B. Dupire and N. Vaz (1988). Cognitive Networks : Immune, Neural and Otherwise, in A. Perelson (Ed.) – *Theoretical Immunology*, Vol.2 SFI – Series on the Science of Complexity – Addison Wesley, New Jersey pp. 359 – 375.

F. Varela and A. Coutinho (1991): Second Generation Immune Network – in *Immunology Today*, Vol. 12 No5 – pp. 159-166.

H. Bersini and F. Varela (1993) : "The Immune Learning Mechanisms: Recruitment Reinforcement and their applications" in *Computing with Biological Metaphors* - Chapman and Hall - R. Patton (Ed.)

V. Detours, Bersini, H., Stewart, J. and F. Varela (1994): Development of an Idiotypic Network in Shape Space - in *Journal of Theoretical Biology* - 170.

J. Stewart (1994) Cognition without neurons: Adaptation, learning and memory in the immune system. *CC AI 11*: pp.7-30.

Calenbuhr, V., Varela, F. and H. Bersini (1996). Immune Idiotypic Network - in *International Journal of Bifurcation and Chaos* - Vol. 6 No 9 - pp. 1691-1702

H. Bersini, and V. Calenbuhr (1996). Frustrated Chaos in Biological Networks - in *Journal of Theoretical Biology*, Vol. 188, No 2, pp. 187-200.

H. Bersini (1999). The Endogenous Double Plasticity of the Immune Network and the Inspiration to be drawn for Engineering Artifacts. In "*Artificial Immune Systems and Their Applications"* - Springer Verlag, pp.22 - 44

P. Matzinger (2002) The Danger Model: A Renewed Sense of Self – In *Science* – Vol. 296. pp. 301 – 305

A. Tauber (2002) The Biological Notion of Self and Non-Self – *Stanford Encyclopedia of Philosophy* – http://plato.stanford.edu/entries/biology-self.

Artificial Immune Systems as Complex Adaptive Systems

Patrícia A. Vargas
DCA-FEEC
Unicamp

Leandro N. de Castro
DCA-FEEC
Unicamp

Fernando J. Von Zuben
DCA-FEEC
Unicamp

Abstract

Complex adaptive systems is a terminology used to describe natural systems, such as biological and social systems, together with their many properties, interactions and resultant emergent behaviours. This paper discusses one general framework to study complex adaptive systems aiming at providing a better understanding and originally demonstrating that artificial immune systems, together with the biological immune system, can be placed in the context of complex adaptive systems. This unique framework is useful for it suggests the possibility of existence of a common language to support the study of the many CAS and artificial intelligence systems. Classifier systems (CS) have been used to model adaptive agents for CAS. By tracing a parallel between AIS and CS, this paper makes it possible to employ AIS as alternative models for CAS agents as well. Novel hybrid and more complex systems are only a few of the outcomes that this paper can bring to the research community.

1 INTRODUCTION

For centuries humankind has been looking into nature with two main goals in mind. First, to pursue the creation of models and explanations of how nature works (e.g., which laws – if they do exist! – govern gravity, the weather, our brains, etc.). Second, humankind has also been looking for inspiration to the development of artefacts to make life easier and more comfortable (e.g., by developing aeroplanes, ships, etc.). With the advent of computers, the human developments inspired by nature supplanted the barrier of the construction of artefacts. It now permeates the more abstract level of computational tools capable of not only simulating the natural world, but also of providing computational solutions to complex problems.

This new perspective of how to use nature as a source of inspiration for the development of computational tools is termed *biologically inspired computing* or *computing with biological metaphors* (Paton, 1994). The other front of using the computer to simulate and better understand the natural world is named *computationally motivated biology* (de Castro & Timmis, 2002).

In order to provide a more general framework with which to study natural systems (e.g., the nervous system, the immune system, the Internet, the social systems, etc.), several research schools have been proposing general theories and a common terminology, such as *complexity*, *emergence* and *adaptive systems*, that embrace features of all natural systems. These theories are very important for they lead to a common language that allows the analysis, interpretation, modelling, and understanding of natural systems under a general and interdisciplinary framework.

From an artificial intelligence perspective, this general framework is a primary pre-requisite for the development of models and computational tools that can be better understood without requiring a great knowledge of a specific domain. In addition, the generality of such a framework can lead to a straightforward development and implementation of systems that combine features of more than one strategy, such as hybrid and ensemble systems.

Complex adaptive systems (CAS) is the terminology adopted here to encompass all these natural systems with their features of diversity, adaptability, and complexity, together with the emergent behaviours that arise from the interactions of their many component parts (Holland, 1995). This paper explores one CAS theory, placing the immune system and artificial immune systems in the broader context of complex adaptive systems.

One of the pioneer computational intelligence algorithms developed to model agents for CAS is the learning classifier system introduced by Holland (1992). We propose the view that artificial immune systems, as introduced by de Castro and Timmis (2002), also serve as models for agents in complex adaptive systems. Indeed, there have been a number of works in the literature comparing specific artificial immune systems (AIS) with learning classifier systems (Farmer *et al.*, 1986; Kauffman, 1989; Varela *et al.*, 1989; Bersini & Varela, 1990; Bersini, 1991; Hunt & Cooke, 1996 and Hofmeyr & Forrest, 2000). However, none of these works thoroughly shows that AIS can be well placed in the CAS framework proposed by Holland (1995).

This paper introduces one of the most popular views of complex adaptive systems (Section 2), and discusses how classifier systems (CS) are used to model agents for CAS (Section 3). This work then provides a brief discussion of the vertebrate immune system, claiming why it can be characterised as a CAS (Section 4). The paper follows with an introduction to a framework to engineer AIS (Section 5), then it explores how this framework is related with classifier systems, providing a survey of the literature that brings together CS and AIS (Section 6).

Not only does this work make the first attempt to place a general-purpose framework to engineer artificial immune systems into the world of complex adaptive systems, but it also opens new avenues of research in the three fields – AIS, CS, and CAS. By presenting this discussion, new developments could be made in the broadest perspective of artificial intelligence, by devising models of CAS, and creating and simulating complex adaptive systems using a framework to design artificial immune systems.

2 COMPLEX ADAPTIVE SYSTEMS

The work proposed by J. Holland (1995) starts with a discussion of how natural (biological and social) systems are formed and self-sustained. These systems are grouped together under the heading *complex adaptive systems* (CAS), in which the (complex) behaviour of the whole is more than a simple sum of individual behaviours. One of the main questions raised is that of how a decentralised – with no central planning – system is self-organised. Note that there is a strong similarity between this concept of a CAS and the concept of *emergent systems* (Holland, 1998). Indeed, complex adaptive systems exhibit emergent phenomena, but this is not the main focus of the discussion to be presented here.

As an instance of a CAS, one can think of the immune system, with its sheer diversity of cells, molecules and organs, all working in concert to provide security against foreign attacks and to aid in sustaining life. Several other examples can be given, such as all bodily systems (e.g., the nervous system, and the endocrine system), insect societies (e.g., ant and termite colonies), trading in commerce (e.g., the stock market). A common aspect here is that there is no central control. Every element composing the system plays its individual role and sometimes adapts itself and interacts with other elements and even systems with the aim of generating and sustaining its integrity and the life of the organism.

Despite the differences among the many complex adaptive systems, in every single case, the persistence of the system relies on three main aspects: 1) interactions of components, 2) diversity, and 3) adaptation. According to Holland (1995), the choice of the name complex adaptive systems is more than a terminology "[i]t signals our intuition that general principles rule CAS behaviour, principles that point to ways of solving attendant problems." (p. 4)

The main objective of (Holland, 1995) is to uncover general principles that will enable the synthesis of refined CAS behaviours from simple laws. The core idea is to develop a well-designed mathematical model for CAS; a formal theory. The steps taken toward this goal were to initially select *seven basics* – four properties (*aggregation*, *nonlinearity*, *flows*, and *diversity*) and three mechanisms (*tagging*, *internal models*, and *building blocks*) – common to all CAS, and then to devise a framework and implement a computer-based model to study CAS. The following is a summary of the intrinsic parts of the Holland's *seven basics*.

2.1 SEVEN BASICS

As described above, Holland's seven basics are divided into four properties: aggregation, nonlinearity, flows and diversity; and three mechanisms: tagging, internal models and building blocks.

2.1.1 Properties

Aggregation in complex adaptive systems occurs in static and dynamic senses. The first sense states how to describe the inherent structure of CAS (a standard way of modelling a CAS), and the second is related to what CAS do aggregate, i.e. how complex large-scale behaviours *emerge* from the aggregate interactions of less complex elements. More precisely, in the first sense, basically, there is an aggregation of categories that afterwards will turn into building blocks for the models. In the second sense, aggregation is a basic characteristic of all complex adaptive systems, where each category aggregates with another category forming a more complex category, thus yielding to more complex hierarchical aggregations.

Non-linearities are present in complex adaptive systems in several distinct levels and they define how non-linear dynamics almost always make the behaviour of the aggregate more complicated. Therefore, the behaviour of a system containing non-linear components is harder to model and to predict.

Flows concern how data (e.g., information, stimuli, electric impulses, resources etc.) propagate through a system and vary over time. It is also divided into two properties, the multiplier effect, which spreads an injected resource or information at a given node or agent throughout the network, and the recycling effect, as the name suggests, helps to maintain the equilibrium in several ways: by adapting the data to a new use or function; by passing it through a cycle again, as for further treatment; or just by starting a different cycle in, i.e. to reprocess.

The last property, *diversity*, is viewed as a necessary feature to generate and maintain a CAS. In fact, perpetual novelty is a hallmark of CAS. It indicates that the diversity is the product of progressive adaptations, as proposed by Charles Darwin (1859), when he observed that the principles of evolution that operated to generate the species, like competition, variation and selection, arise from the diversity of species.

2.1.2 Mechanisms

The first mechanism, denoted *tagging*, refers to tag-based interactions (labelled interactions, i.e. identified and/or classified) that provide a sound basis for filtering, specialisation, co-operation, competition, formation of aggregates, manipulation of symmetries and for selective interactions.

The *internal models* mechanism is the expression chosen to refer to mechanisms for anticipation (the act of considering something beforehand, i.e. foreknowledge) and prediction (the act of reasoning about future events or

possibilities, especially on the basis of special knowledge, i.e. foresight). In fact, internal models distinguish CAS from other complex systems. They balance exploration with exploitation (Holland, 1995), providing the make of careful systematic searches of profitable and useful resources. There are two kinds of internal models, *tacit* - that simply prescribes a current action and *overt* - that explores alternatives (looks ahead), allowing inferences.

The last mechanism, named *building blocks* or *generators* constitute basic elements or parts that compose internal models. In fact, relevant building blocks are combined to model new situations, therefore, to generate internal models or a completely novel CAS.

3 CLASSIFIER SYSTEMS TO MODEL AGENTS FOR COMPLEX ADAPTIVE SYSTEMS

The framework proposed by Holland (1995) for developing adaptive agents for CAS was introduced as consisting of three major built-in stages:

1) In the *performance system* stage, agents are viewed and described as a collection of message processing rules. The syntax of the rules depends on their interaction with the environment. A set of detectors and a set of effectors manage this system-environment interaction. Additionally, the *performance system* specifies the agents' capabilities at a fixed point in time and it prepares these agents to novel situations without having all rules a priori.
2) In the *credit-assignment* stage the core idea is to provide the agents with the capability of adapting to the environment. In the performance system, a number of rules can be fired simultaneously according to the interactions of the system. As a consequence, these rules must compete with one another in order to have a single rule being selected to determine the output of the system. Each rule has a *strength* assigned, which is modified via *credit-assignment* on the basis of experience (e.g., a Bucket Brigade algorithm (Booker *et al.*, 1989)). Credit-assignment is performed in response (reward- reinforcement or punishment) to a 'payoff' received from the environment.
3) The *rule-discovery* stage describes another way of endowing agents with adaptability by allowing the system to automatically generate 'plausible' rules. It should be done always taking the past experience into consideration. The author uses the notion of schemas (likened to building blocks) and genetic algorithms as tools for rule discovery.

The framework described above for modelling CAS' agents is reminiscent of classifier systems. Indeed, classifier systems have already been used to model CAS (Holland, 1992).

3.1 CLASSIFIER SYSTEMS

J. Holland proposed the principles of classifier systems in 1976 (Holland, 1992). They constitute an evolutionary computation strategy for creating and updating rules (named *classifiers*), that are able to point out suitable actions for adaptive agents in changing environments. In some cases, classifiers are kept under permanent evolution to improve their performance. There are some basic concepts related to classifier systems:

1) Classifiers are composed of an antecedent part (*condition*) and a consequent part (*action*). The antecedent part of a classifier is a string of fixed size composed of elements of the ternary set {0,1, #} - the symbol "#" is called "don't care", which can assume any value over a pre-defined finite alphabet (e.g., a wild card in a playing game of cards).
2) A classifier system communicates with the environment through his message *detectors* and *effectors*. Detectors receive and decode messages from the environment. Effectors propose actions on the environment (Figure 1).
3) A *strength* is associated with each classifier, and expresses the energy or power of the classifier during the evolutionary process.
4) *Feedback* from the environment defines an appropriate reward to the active classifier, proportional to the quality of the action(s) it proposes.
5) The *specificity* of a classifier is inversely proportional to the quantity of don't care symbols (#) in its antecedent part. Classifiers with low specificities can match larger numbers of messages from the environment, and vice-versa.

Classifier Systems are divided into three interactive and distinct sub-systems: the *Rule and Message Sub-System*, the *Apportionment of Credit Sub-System* and the *Rule Discovery Sub-System* (Figure 1).

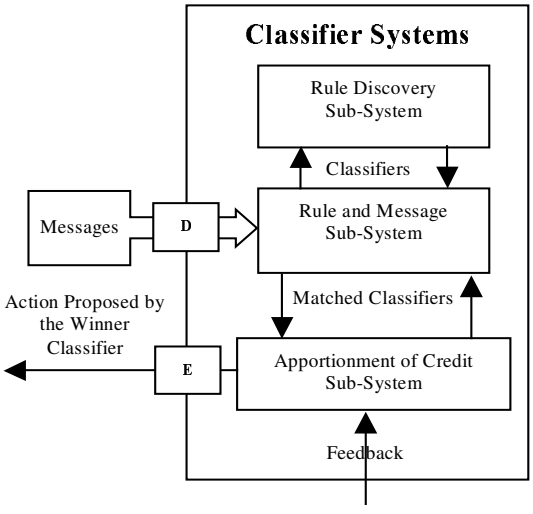


Figure 1: Simplified flow (Classifier Systems ↔ Environment). D: detectors; E: effectors.

The *Rule and Message Sub-System* decodes messages in a way that the classifier system can recognise them. Then, all classifiers try to match their antecedent part with the message (comparison phase). This matching can be made by bit-to-bit comparison, according to specific rules, or just by calculating a variation of the Hamming distance.

START OF THE EVOLUTIONARY PROCESS
START OF ONE EPOCH
Step 1: Detect environmental messages.
Step 2: Code messages.
Step 3: Select matched classifiers.
Step 4: Begin Competition:
- Calculate each competitor's bid.
- Point out the winner.
- Collect taxes from competitors and the winner.
Step 5: Act on the environment.
Step 6: Receive environment's feedback.
Step 7: Reward or not the winner classifier.
Step 8: Charge life tax from all classifiers.
Step 9: If it is not the end of an epoch, go to *Step 1*.
END OF ONE EPOCH
Step 10: Select classifiers to apply genetic operators (generally, the more adapted ones).
Step 11: Apply crossover and mutation operators to generate the offspring.
Step 12: Select classifiers to be replaced (generally the less adapted ones).
Step 13: Insert the generated offspring of *Step 11* into the population, to replace the classifiers selected on *Step 12*.
Step 14: If it is not the end of the evolutionary process, go to *Step 1*.

Figure 2: Basic algorithm of a Classifier System.

Each classifier that matches the environmental message is sent to the *Apportionment of Credit Sub-System* where they participate in a *competition*. This competition is a random process biased by the strength of classifiers that matched their antecedent part with the message - classifiers with higher strengths have higher probabilities of winning the competition. The winner will act on the environment. The environment will reply in response to the action proposed by the winner classifier. The *Apportionment of Credit Sub-System* incorporates a value generated by this feedback from the environment to the strength of the active classifier at that moment. Once the feedback is received from the environment and the credit is attributed to the winner classifier, a new message will be provided by the environment, describing its new state.

The process goes on iteratively, epoch by epoch. At the end of each epoch, the classifier system will take part in another process of evolution at the *Rule Discovery Sub-System*, where genetic operators are applied to produce a new generation of classifiers. Basically, a genetic algorithm chooses classifiers with large strengths and promotes reproduction among them by applying the genetic operators of crossover and mutation. Offspring replace weak individuals (the ones with lower strength).

Several taxes are collected from all individuals in the population of classifiers. A *life tax* is charged from each classifier at each iteration. A *bid tax* is collected from each participant of the competition. The winner in the competition also pays a tax for the right to act on the environment. A simplified algorithm is depicted in Figure 2. It provides the starting point from which one can have a better insight of the internal processes of classifier systems.

Based on previous studies, we can state that classifier systems constitute a sufficiently flexible tool for self-adaptation to time-varying contexts. Additionally, they have shown effectiveness on the production of secondary responses to previously presented stimuli and are able to react promptly to changes in the environment due to their diversity preservation feature (Vargas *et al.*, 2002).

Note that the basic concepts of a classifier system are reminiscent of the framework, discussed previously, for adaptive agents to model CAS. For instance, the set of detectors/effectors plus the IF/THEN rules (classifiers) correspond to the performance system. The rule discovery and credit apportionment systems are equivalent to the rule discovery and credit assignment algorithms, respectively, as summarised in Table 1.

Table 1: Mapping a classifier system into the Holland's framework for adaptive agents to model CAS.

Classifier Systems	**Adaptive Agents for CAS**
Set of detectors/effectors plus the classifiers	Performance system
Rule discovery system	Rule discovery system
Credit apportionment system	Credit assignment system

4 THE IMMUNE SYSTEM AS A COMPLEX ADAPTIVE SYSTEM

It is possible to identify the seven basics of all CAS in the mammalian immune system.

4.1 PROPERTIES

1) *Aggregation* – In the immune system, aggregation can also be divided into *static* and *dynamic*. The static aggregation corresponds to the intrinsic structure of the

immune system, which is composed of a large variety of cells. Among them, lymphocytes are the most important ones, from a biological and computational perspective. These cells can be naturally categorised according to their physiology and function, mainly into B- and T-cells. They all act together to protect our bodies against foreign attacks by pathogens, and against malfunctioning self-cells. The physiology and functions that lead to an aggregation of immune cells can be easily exemplified. For instance, those naïve lymphocytes that mature within the bone marrow are termed B-cells, and those that mature into the thymus are named T-cells. Macrophages are different from B- and T-cells for they act by scavenging infected cells and other debris found in the blood stream and lymph. Interactions among cells and molecules in the immune system are not only abundant, but also necessary for its functioning. Without the help of T-cells, B-cells cannot detect pathogens hidden inside and causing damage to our own cells. Also, chemical products released by B- and T-cells stimulate and signal to other cells, such as macrophages and even other B- and T-cells, to 'detect' pathogens and perform their roles against them. Therefore, the *dynamic aggregation* (interaction) in the immune system leads to more powerful immune responses to pathogens. Aggregation is thus pervasive in the immune system.

2) *Non-linearities* - In the case of the immune system, there are, among others, the effects of saturation in antibody production and lymphokine secretion. Regardless of the amount of pathogens invading the organism, antibody production and lymphokine secretion cannot raise above a certain level. Nonlinearity is also evident when one considers that, for example, two antibodies acting together have a different effect in an antigen than if we sum the effect of both of them acting individually. This holds true for all elements in the immune system, from lymphokines to macrophages.

3) *Flows* - Alike nonlinearity, several levels of flow can be identified in the immune system. From the flow of immune cells throughout the organism, to the flow of their secreted chemicals (e.g., lymphokines). The multiplier effect can be observed during B- and T-cell clonal expansion (proliferation) in response to antigenic stimuli. When an antigen is detected, some lymphocytes are selected due to an antigenic recognition and start proliferating, thus releasing (spreading) antibodies and lymphokines in the lymph and blood stream. An instance of the recycling effect is the affinity maturation process, which allows immune receptors to become more adapted to the antigenic stimuli. Another example is the release of lymphokines, mainly by T-cells, signalling the death of an antigen and thus ending the immune response allowing the immune system to return to its equilibrium (non-responsive) state.

4) *Diversity* - In the immune system, B-cells, T-cells, macrophages, granulocites, chemokines (lymphokines), etc., all contribute to a suitable immune functioning. Diversity in the immune system can also be studied in different levels. For instance, at an aggregation level, there are different types of cells (e.g., B-cells, T-cells and macrophages), molecules (e.g., antibodies, and lymphokines), and organs (e.g., bone marrow, thymus, and lymph nodes) composing the immune system. In addition, many of these groups have an intrinsic diverse set of components. For example, it is known that there is a large variety of lymphocyte receptors that endow the immune system with the capability of recognizing an even more diverse set of antigenic patterns.

4.2 MECHANISMS

1) *Tagging* - Each immune cell has its particular design; not a single element is the perfect copy of another one. Nevertheless, all elements of a given type (e.g., B-cells) share some common features (tags) that allow them to be categorised as B-cells. The same is true for other cell types and molecules.

2) *Internal models* - When the immune system is primed with a type of pathogen, it builds a repertoire of cells and molecules that is specialised in recognising this type of pathogen. The immune system thus builds an internal model that allows it to anticipate a known antigenic patern, thus promoting a faster recognition and elimination of previously seen pathogens. The idea of internal models in the immune system is largely studied in theories of immune networks. In the original immune network theory, introduced by Jerne (1974), individual cells and molecules are capable of recognising each other and antigens as well. As an outcome, the immune system naturally generates and maintains a network of immune cells and molecules that interact with each other even in the absence of external stimuli. The same immune cell that can recognise another immune cell can also recognise an antigen. The immune cell recognised has similar attributes to the antigen and is thus called an *internal image* of the antigen. The immune cells and molecules that are currently available (available repertoire) in the immune system can be likened to the tacit internal models, whilst those cells that are constantly being created and replacing the existing low stimulated ones can be likened to the overt internal models.

3) *Building blocks* - A clear example of the presence of building blocks in the immune system is the use of genes, selected from gene libraries, to construct lymphocyte receptor molecules. Individual genes and the libraries themselves can be considered as building blocks to generate receptor molecules.

5 A FRAMEWORK TO ENGINEER ARTIFICIAL IMMUNE SYSTEMS

As with any new field of research, the various works on artificial immune systems lack a more fundamental set of ideas, mechanisms, and common language for their de-

scription, understanding and development. To alleviate these disparities, a first textbook in English is now being released (de Castro & Timmis, 2002). This section briefly reviews and discusses the framework proposed in this book.

One of the contributions of this textbook is the proposal of a generic framework with which to design and understand artificial immune systems. Artificial immune systems have been defined as adaptive systems inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving (de Castro & Timmis, 2002). To design an AIS, the authors proposed a layered framework with three main parts:

1) a *representation* for the components of the system,

2) a set of *mechanisms to evaluate the interactions* of individuals with the environment and each other, and

3) some *adaptation procedures*.

Within each of these layers there are various strategies. For instance, in layer 1 – representation – shape-spaces (Perelson & Oster, 1979) play a major role. A shape-space may be understood as a (search) space where attribute strings are used as abstract models to represent immune cells and molecules. The idea is to use the attribute strings as means of quantifying the degree of recognition (affinity) between immune cells, and between them and the environment.

In layer 2 – interactions – the environment may be simulated by a set of input stimuli, and fitness and/or affinity functions. These allow the determination of the (relative) quality of the individuals composing the population. For example, if using binary strings in layer 1 to represent the immune receptors, a metric such as the Hamming distance is a candidate to be used to quantify the degree of similarity or dissimilarity (recognition) between two bitstrings.

The procedures of adaptation in layer 3, that govern how the behaviour of the system varies over time, are usually simplified models of an immune function, process, or theory. For instance, clonal selection (Burnet, 1959), negative selection (Kruisbeck, 1995), immune networks (Jerne, 1974), and so forth, have been largely used by the AIS community. These procedures usually take the affinity/fitness measures of layer 2 as some of their inputs.

5.1 AIS FRAMEWORK IN A BROADER PICTURE

Note that this framework is not much different from the one proposed by Michalewicz and Fogel (2000) when describing heursitic problem solving techniques. They suggest that three main concepts are involved in problem solving: *representation*, the definition of an *objective function*, and the choice of an *evaluation function*. Additionally, the proposed framework for AIS also brings some similarities with the framework introduced by Holland (1992) to model adaptation in natural and artificial systems. Holland suggests that such a framework might be composed of an *environment* undergoing adaptation, an *adaptive plan* which determines successive structural modifications in response to the environment, and a *performance measure* of different structures in the environment.

There are also many similarities between the proposed AIS framework and some basic design principles of other biologically inspired techniques, such as neural networks and evolutionary algorithms. Artificial neural networks require a model for an artificial neuron and a network structure (representation), one or more activation function (interactions), and a learning algorithm (adaptation procedure). Evolutionary algorithms also require some sort of data structure (representation), fitness function (interactions), and variation (genetic) operators to be used in the adaptation procedures.

Therefore, though there might be slight differences among the many frameworks for problem solving and modelling of complex adaptive sytems, it is possible to stress a set of basic components (building blocks) from which some of them may be useful in a particular context. As examples, one may stress an environment in which the systems are built, a given representation scheme for individuals that inhabit the environment, some evaluation mechanisms to allow for a qualitative distinction of individuals, and adaptation strategies to change the configuration (state) of the system.

6 ARTIFICIAL IMMUNE SYSTEMS AND CLASSIFIER SYSTEMS: A SURVEY AND A COMPARISON

There have been a number of works in the literature comparing specific artificial immune systems with classifier systems. Among others, one can highlight the works of Farmer *et al.* (1986), Kauffman (1989), Varela *et al.* (1989), Bersini & Varela (1990), Bersini (1991), Hunt & Cooke (1996) and Hofmeyr & Forrest (2000) (see Table 1). This section aims at surveying the works from the literature that liken specific AIS with classifier systems (CS), and placing the framework for engineering AIS, introduced in Section 5, in the context of classifier systems.

6.1 A SURVEY OF CS AND AIS

To date, all the works comparing AIS with CS were made under specific application scenarios. To the authors' knowledge, the first work to look for similarities and differences between both systems was presented by Farmer *et al.* (1986). Basically, the authors wrote both systems in the form of a dynamical system, resulting in equations of motion to describe their dynamics. A CS was used to model the immune system (IS) by drawing an analogy between individual classifiers and antibodies. They pointed out that the main difference between both

systems was the nature of the nonlinearity in both equations. They also stressed other differences, like the interaction with the external environment and the system of message passing used in the classifier system. Additionally, the authors even stated that "It is an accident that there is any similarity at all between both systems". Despite that, some similarities were presented. They found out that the generation of new solutions acts in precisely the same manner in both systems (providing creativity), and that both frameworks are strongly non-linear dynamical systems. Unfortunately their work had relatively little impact on CS research, but is considered a landmark of the field of AIS (de Castro & Timmis, 2002).

Table 2: A synthesis of the comparisons performed by [1] Farmer *et al.* (1986) and [2] Hofmeyr & Forrest (2000).

Classifier System	Work	Artificial Immune Systems
Classifier	[1]	Antibody
Specificity	[1]	Specificity
Condition	[1]	Epitope
Tax	[1]	Dissipation term
Payoff	[1]	Antigen reduction
Economy	[1]	Concentration update rule
Performance function	[1]	rate of antigen removal
External message	[1]	Antigen
	[2]	Detector
Message list	[1]	Paratopes and antigens
	[2]	Network traffic
Action	[1]	Paratope
	[2]	Isotypes
Strength	[1]	Concentration
	[2]	Immature, mature, activated and memory states
Genetic operators	[1]	Genetic operators
	[2]	Random detectors
Matching	[2]	via the r-contiguous bits rule
Competition	[2]	Bidding for messages
More specific match wins	[2]	more specific match wins
Support	[2]	Activation threshold
Message intensity	[2]	Sensitivity level
Bucket brigade	[2]	Affinity maturation
Triggering	[2]	Negative selection

For Bersini and Varela (1990) the Immune System is more like Holland's classifier system (either escape 'brittleness' (fragility) or 'semantic closure'). The 'problem solving' qualities belong to an evolving, adaptive and self-organising population of interactive operators. The authors suggest that a complete comparison between CS and an immune network model covers the whole cognitive domain: search, adaptability, memory and learning.

In Hunt and Cooke's (1996) point of view, their immune network model combines the advantages of learning classifier systems with some of the advantages of neural networks, machine induction and case-based retrieval. The authors believe that although their AIS has similarities with both systems, it differs from both of these in a number of significant aspects. These differences have the potential to make their AIS applicable in situations where neural networks or learning classifier systems are unsuitable, e.g. learning classifier systems find it difficult to deal with problems which lack separation between global solutions or have many locally optimal rules. This is not the case for their AIS.

Hofmeyr and Forrest (2000) referred to an AIS for network intrusion detection as a resemblance to the architecture of a classifier system. The mapping between their AIS and CS was not 1 to 1. In their implementation nothing corresponded to the action part of a classifier. Furthermore, the authors were the first ones to suggest that their AIS could be added to the repertoire of CAS.

Table 2 summarises the comparisons made by Farmer *et al.* (1986) [1] and Hofmeyr and Forrest (2000) [2] with classifier systems. Note that, in some cases, an equivalence between both works can also be drawn.

6.2 CS AND A FRAMEWORK TO DESIGN AIS

The comparison of specific AIS with classifier systems can be extended to a more general comparison of AIS in the light of the framework described in Section 5 as follows (see Table 3).

Classifiers may correspond to attribute strings, in a given shape-space, representing immune cells and molecules. These strings can be simple structures such as binary strings or more complex structures such as one containing symbolic values. The communication with the environment is performed via a set of input stimuli, or one or more fitness/affinity measures. A detector in a classifier system corresponds to a receptor in an immune cell, and the effector might be likened to lymphokines excreted by the immune cells. Other types of effectors can also be available in an AIS depending on their rationale (e.g., the elimination or classification of given pattern). The strength of a classifier might correspond to the affinity value of a given immune cell or molecule, which in turn will be responsible for determining an action of or to be acted upon this cell or molecule. Though most AIS do not employ wild cards (don't cares) in their representation, it can also be found in the literature (e.g., Hart *et al.*, 1999).

Table 3: General comparison between AIS and CS.

Classifier Systems	Artificial Immune Systems
Classifiers	Attribute strings
Strength of a classifier	Fitness/affinity
Detector	Receptor
Effector	Lymphokines excreted
Don't cares (#'s)	Don't cares

In the highest level it is also possible to link AIS with classifier systems. The rule and message sub-system can be likened to the set of attribute strings representing immune cells and molecules. The apportionment of credit sub-system can be equated to the set of mechanisms to evaluate the interactions of individuals with the environment. Finally, The rule discovery sub-system corresponds to the procedures of adaptation for AIS. Table 4 summarises the framework comparison between AIS and classifier systems.

Table 4: Framework comparison between AIS and CS.

Classifier Systems	Artificial Immune Systems
Rule and message sub-system	Representation (shape-spaces)
Apportionment of credit sub-system	Mechanisms to evaluate interactions
Rule discovery sub-system	Procedures of adaptation

In addition to the general comparison made above, there are some equivalencies between both systems that should be emphasised:

- both systems employ tags in their inside and outside interactions;
- it is possible to observe an intrinsic willingness to combat or to compete;
- the matching processes are isomorphic in function;
- both systems are frequently updating their internal models.

As an additional contribution, we will follow Holland's suggested common representation, described in Section 4, in order to create a simplified model for the AIS.

There are three stages that we must follow to start the modelling (see Section 4 for details). The first one is the *performance system*. In this stage agents are viewed and described as a collection of message processing rules. The exposition is going to start by identifying the agents in an immune system. Recalling that agents are active elements that form CAS, it is possible to say that the immune system has several agents. Among them, choose the antibodies as the representative agents of the immune system. Each agent can be modelled as a classifier, i.e. a message-processing rule. The rule syntax will vary according to the interaction with the environment. The agents will have detectors, to detect environmental stimuli, e.g. tags on antigen surfaces, and effectors to indicate the next step or what task should be accomplished.

In stage 2, it is necessary to define the credit assignment. This can be done by taking into consideration the affinity measure between the antibody and the antigen, i.e. the classifier and the stimuli.

Finally, the last stage is Rule discovery and it can be dealt with a modified genetic algorithm, taking the past experience of memory cells (classifiers with high strength) into consideration.

7 FINAL REMARKS

This paper reviewed complex adaptive systems as all natural systems exhibiting a number of diverse interactive individuals and presenting adaptability. It was the first attempt to place artificial immune systems into the context of complex adaptive systems proposed by J. Holland (1995).

A framework to engineer artificial immune systems and the main concepts involved in the design of classifier systems were also presented. These allowed a survey and general comparison between CS and AIS. As both systems were demonstrated to be equivalent and classifier systems have been used to model agents for CAS, AIS are also suitable to model agents for complex adaptive systems.

The framework to study CAS is generic in the sense that it provides a common language for the modelling, understanding, simulation, and creation of computer models for CAS. This common language, together with the suitability of AIS to study CAS, lead to a broader applicability of AIS and to its possible combination with several techniques inspired by other systems or processes. This work therefore opened new avenues of research that may result in new developments of artificial intelligence approaches, through the proposal of models of CAS and the creation and simulation of complex adaptive systems by using a framework to design artificial immune systems.

Indeed, de Castro and Timmis (2002) have surveyed a number of works in the literature describing and proposing hybrids of AIS with neural networks, evolutionary algorithms, fuzzy systems, and so forth.

Acknowledgements

Patrícia A. Vargas would like to thank CAPES for the financial support.

Fernando J. Von Zuben would like to thank CNPq (ns.: 300910/96-7 and 52.1100/01-1) for the financial support.

Leandro N. de Castro would like to thank CNPq (Profix n. 540396/01-0) for the financial support.

References

Bersini, H. (1991). Immune Network and Adaptive Control. Proceedings of the First European Conference on Artificial Life, MIT Press, pp. 217-226.

Bersini, H. & Varela, F. (1990). Hints for Adaptive Problem Solving Gleaned from Immune Networks. Proceedings of the First Conference on Parallel Problem Solving from Nature, pp. 343-354.

Booker, L. B., Goldberg, D. E. & Holland, J. H. (1989). Classifier Systems and Genetic Algorithms. Artificial Intelligence, vol. 40, pp. 235-282.

Burnet, F. M. (1959), *The Clonal Selection Theory of Acquired Immunity*, Cambridge University Press.

Darwin, C. (1859), *On the Origin of Species by Means of Natural Selection*, 6th Edition, [Online Book] www.literature.org/authors/darwin.

de Castro, L. N. & Timmis, J. I. (2002). *Artificial Immune Systems: A New Computational Intelligence Approach*, Springer-Verlag.

Farmer, J. D., Packard, N. H. & Perelson, A. S. (1986). The Immune System, Adaptation and Machine Learning, Physica 22D, pp. 187-204.

Hart, E. & Ross, P. (1999), The Evolution and Analysis of a Potential Antibody Library for Use in Job-Shop Scheduling, in *New Ideas in Optimization*, D. Corne, M. Dorigo & F. Glover (eds.), McGraw Hill, London, pp. 185-202.

Hofmeyr, S. A. & Forrest, S. (2000). Architecture for an Artificial Immune System, Evolutionary Computation, 8(4), pp. 443-473.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems*. MIT Press.

Holland, J. H. (1995). *Hidden Order: How Adaptation Builds Complexity*. Addison-Wesley, Inc.

Holland, J. H. (1998). *Emergence: From Chaos to Order*. Addison-Wesley, Inc.

Hunt, J. E. & Cooke, D. E. (1996). Learning Using an Artificial Immune System, Journal of Network and Computer Applications, 19, pp. 189-212.

Jerne, N. K. (1974), Towards a Network Theory of the Immune System, Ann. Immunol. (Int. Pasteur) 125C, pp. 373-389.

Kauffman, S. A. (1989). Principles of Adaptation in Complex Systems, in D. Stein (ed.), *Lectures in the Sciences of Complexity*, Addison Wesley.

Kruisbeek, A. M. (1995), Tolerance, *The Immunologist*, 3/5-6, pp. 176-178.

Paton, R. (ed.) (1994), *Computing with Biological Metaphors*, Chapman & Hall.

Perelson, A. S. & Oster, G. F. (1979), Theoretical Studies of Clonal Selection: Minimal Antibody Repertoire Size and Reliability of Self-Nonself Discrimination, *J. theor.Biol.*, 81, pp. 645-670.

Varela, F., Sanchez, V. & Coutinho, A. (1989). Adaptive Strategies Gleaned from Immune Networks in B. Goodwin and P. Saunders (eds.), Evolutionary and Epigenetic Order from Complex Systems: A Waddington Memorial Volume. Edinburgh U. Press.

Vargas, P. A., Lyra, C. & Von Zuben, F. J. (2002). Online Approach for Loss Reduction in Electric Power Distribution Networks Using Learning Classifier Systems, in *Lecture Notes in Artificial Intelligence* (LNAI 2321), Springer-Verlag, pp. 181-196.

Building a Robust Distributed Artificial Immune System

Johan Kaers[1], **Richard Wheeler**[2] and **Herman Verrelst**[1]
[1]Data4s Future Technologies
Ambachtenlaan 13G, 3001 Heverlee, Belgium
{johan.kaers, herman.verrelst}@data4s.com
[2]Public Voice Lab
Opperngasse 24, Vienna, Austria, rew@pandora.be

Abstract

It is shown how to build and use large-scale networks of communicating Artificial Immune Systems that are robust against random failures or intentional attacks. The nodes of the network graph contain the immune systems while the edges are channels over which antibodies are circulated. Because there's a constant flow of antibodies through the nodes, the anomaly detection performance increases over time. This *Distributed Artificial Immune System* model (or DAIS) is introduced and analysed for networks with ring shaped, fully connected and finally arbitrary topologies. These results are combined with recent results from Complex Network Theory about the construction and robustness of random (Erdős-Rényi) and scale-free (Barbási-Albert) graphs. We show that by using these kinds of graphs, it can be guaranteed that the DAIS will degrade gracefully when nodes fail and will maintain a reliable detection performance, even under very high failure conditions. The mapping of the model onto a real-world software architecture and it's possible usages are also discussed.

1 Introduction

The *Artificial Immune System* (AIS) model introduced in [9] is an inherently distributed anomaly detection technique. The main reason for this is that a collection of independent and atomic detection particles or *antibodies* are used to determine whether the system under consideration has changed. It was soon observed [9] that this allows using a collection of artificial immune systems that each contain part of the antibodies. In [11] a similar type of AIS model is introduced (as ARTIS) and applied to network intrusion detection under the name LISYS. Again, the anomaly detection nodes form a distributed system, each containing part of the antibodies and observing the same environment (a LAN). [12] investigates a distributed Artificial Immune Model, also applied to network intrusion detection. There, the evolutionary processes that drive the model can trigger an exchange of antibodies between its sub-systems.

Here the detection performance of a network of Artificial Immune Systems is analysed while they exchange antibodies. The nodes of the network graph contain the immune systems while antibodies are circulated over the graph's edges. This greatly increases the number of unique antibodies encountered at any given node over time and hence also the detection performance. If the network graph is constructed well, soon all antibodies in the system will have passed through all nodes. It has the advantage that all anomaly detection knowledge is shared over the entire system. If an advanced immune response model is used, like in [10] or [12], products of the involved mechanism (e.g. memory detectors) could be communicated throughout the system with the same ease. The goal is to find a type of graph that is robust against random failure or intentional attack on it's nodes and at the same time reasonable in the amount of resources it consumes (e.g. the number of edges it has, the number of antibodies it holds). In section 2 the DAIS model is formally defined and analysed for the ring-shaped and fully-connected topologies. The fact that neither realize this goal, leads us to a consideration of *Complex Network Theory*.

Recently it has become clear that complex network theory, with roots in graph theory and statistical mechanics, can explain the behaviour of a wide variety of biological [16], physical [3], sociological [14] and technological systems [17], ranging from the World Wide Web [1] to movie actor collaboration networks.

In addition, it also shows how to build networks (graphs) that are robust against node failure or attacks. Combining the Distributed Artificial Immune System model with this holds the promise of building a large scale distributed pattern recognition system that degrades gracefully under failure conditions. As the analysis in section 3 will show, using this type of graph creates a system that satisfies both our design goals. It is robust against failure and attacks on the nodes because fragmentation of the graph does not occur unless under extremely high failure rates. And while reaching almost the same detection performance as a fully connected graph with the same number of nodes, it needs far less edges.

The possible applications of this model are discussed in section 5 as well as the mapping onto the software implementation we used for the experiment included in this report. Finally, some conclusions and possible tracks for future work are outlined.

2 The DAIS

Throughout the paper we use the Distributed Artificial Immune System model defined in this section. It extends the Artificial Immune System model first introduced in [9] by embedding a number of identical AISs in the nodes of a graph. The edges connecting the graph's nodes are communication channels over which antibodies are circulated. A discrete time dimension is added, at every time-step all nodes send a fraction of their antibodies to their neighbouring nodes in the graph. All nodes are identical in the sense that they share the same set of *self strings*, match-rule, size of the antibody set and antibody migration probability. Because of the randomness in the antibody generation algorithm that is used, the actual antibodies present in the nodes do differ. Also, the environment that is monitored and from which the *antigens* originate can be different at every node. During a series of time-steps the antigens are matched against all antibodies that pass through a node to determine if they differ from the pre-defined self set.

2.1 The DAIS Model

More formally, the set $N = \{n_1, \ldots, n_{N_n}\}$ contains the nodes of the network graph. $E \subseteq N \times N$ contains the pairs of nodes that are connected. The graph $G_t =< N, E >$ defines the *network topology* of the DAIS. The set M_i contains the neighbours of the node n_i.

$$\forall i = 1 \ldots N_n : M_i = \{n_j \in N \mid (n_i, n_j) \in E\} \quad (1)$$

The *degree* k of a node is the number of neighbours it has, $k_i = |M_i|$

All nodes contain an identical artificial immune system with the following properties

- **Self Set**
  It detects changes in a *Self Set* S, consisting of N_s binary strings of l bits.
- **Match-Rule**
  The contiguous bit matching rule, with match-length r is used. The probability of a match between 2 random string is [9].

$$p_m \approx 2^{-r}((l - r)/2 + 1) \quad (2)$$

- **Antibody Set**
  An algorithm such as the ones found in [7] is used to generate an *Antibody Set* A with N_a *Antibodies*.
- **Antibody Migration**
  At every time-step an antibody has a probability p_{am} of migrating to a node neighbouring the one it is currently found in. The destination node is also selected at random.
- **Antigen Set** Every node has an *Antigen Set* that contains N_g binary string and captures the current state of the environment. In contrast to the self set, this set may differ from node to node. The antigens bit-string are matched over time against the antibodies passing through the node to determine if they differ from the Self Set bit-strings.

Using these notations we can determine the number of time-steps needed to ensure that an arbitrary non-self antigen will be detected with some predetermined probability. This depends on the total number of distinct antibodies that have been present in the node while the antigen was matched against them. Therefore, the key is to understand the *Antibody Migration* behaviour in the graph.

2.2 The Ring Shaped DAIS

As a first example, we instantiate the model for a ring-shaped network. In this topology, every node has exactly 1 neighbour, and they form the only cycle in the graph. So, $M_i = \{N_{i+1}\}$ for $i < N_n$ and $M_{N_n} = \{N_0\}$, $k_i = 1$.

The crucial observation is that the probability $p_d(t)$ for an antibody to migrate over d nodes in time t follows the *binomial distribution* with parameters t and p_{am}. Because at every time-step it has a chance p_{am}

of migrating to the next node and the time-steps are independent.

Using the notation

$$C_k^n = \left(\begin{array}{c} n \\ k \end{array} \right) = \frac{n!}{k!(n-k)!}$$

$$p_d(t) = C_d^t \; p_{am}^d \; (1 - p_{am})^{t-d} \qquad \forall t \geq d \qquad (3)$$

This means that for a random node in the ring, at time t, the number of antibodies that passed through it starting from the node d edges away is

$$N_a \sum_{j=d}^{t} C_j^t p_{am}^j (1 - p_{am})^{t-j} \qquad (4)$$

So, for the entire ring, the total number of distinct antibodies encountered at the node after time t is equal to

$$N_a(t) = N_a[1 + \sum_{i=1}^{N_n - 1} \sum_{j=i}^{t} C_j^t p_{am}^j (1 - p_{am})^{t-j}] \qquad (5)$$

Assuming a self-set of random binary string, the non-self detection performance is [9]

$$p_d \approx 1 - e^{-p_m N_a(t)} \qquad (6)$$

which is reached at time-step t where

$$N_a(t) \approx -\frac{\ln(1 - p_d)}{p_m} \qquad (7)$$

2.3 The Fully Connected DAIS

From the perspective of building a robust network, the ring-shaped topology example is far from optimal. With *robustness* we mean tolerance of the graph's topology and other properties against errors, be it the failure of nodes in the graph or the removal of edges between them. After only 2 nodes are removed, it breaks up in 2 disconnected parts. Removing even more nodes creates more disconnected chain-like graphs of rapidly decreasing size.

Going to the other extreme, instead of connecting every node to only 1 neighbour, we can connect all nodes to all other nodes. So, $M_i = N \backslash \{N_i\}$ and $k_i = N_n - 1$.

In this configuration, for any node, at any time-step, the chance that a random antibody will migrate to it is

$$p_n = \frac{p_{am}}{N_n - 1}$$

The probability that an antibody has visited a node k times at time-step t follows the binomial distribution with parameters p_n and t. So, at time-step t an antibody has been at a node once or more times with probability

$$\sum_{i=1}^{t} C_i^t p_n^i (1 - p_n)^{t-i}$$

And since all N_n nodes contain N_a antibodies, the total number of distinct antibodies encountered at any node after time t is equal to

$$N_a(t) = N_a[1 + (N_n - 1) \sum_{i=1}^{t} C_i^t p_n^i (1 - p_n)^{t-i}] \qquad (8)$$

Since all pairs of nodes are connected to each other, this kind of topology is maximally robust against node failures. But the number of edges grows quadratic in the number of nodes. This soon becomes unrealistic for large networks because in actual applications an edge corresponds to a communication channel that consumes resources of some kind (e.g. bandwidth on a LAN).

3 Complex Network Theory

Keeping the above examples in mind, we would like to find a type of graph that is tolerant against removal of nodes and at the same time, its number of edges shouldn't grow too fast with the number of nodes. Also, we would like to keep the ability present in the fully connected graph to travel between any 2 nodes in a small number of steps.

3.1 Graph Theory

Before using the graphs from complex network theory we need to state some concepts from graph theory.

- A graph is *connected* if there is a *path* between any two nodes, if there is at least one way to go from any node to any other node by following the edges of the graph.
- The *path length* between two nodes is the minimum number of edges that have to be followed between the nodes. The *diameter* of a graph is the maximum path length between any two nodes.
- A *small world* graph has, despite it's large number of nodes, a relatively short path between any two nodes.

- The *degree distribution* of a graph captures the spread in the number of edges a node has. The probability distribution function $P(k)$ gives the probability that a randomly selected node has exactly k edges.

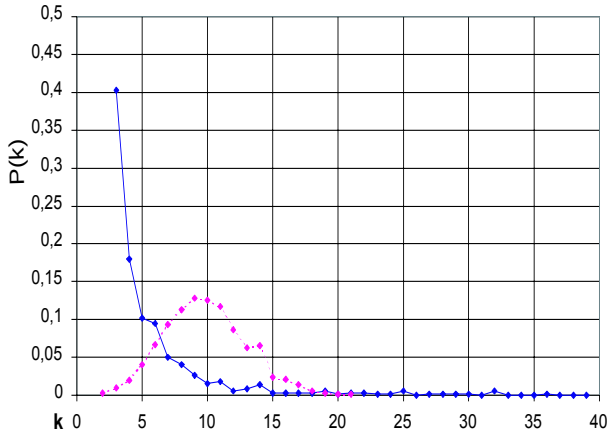


Figure 1: Degree Distribution of an Barabási-Albert (solid) graph of 1000 nodes with $m = 3$ and an Erdős-Rényi graph (dashed) with $p = 0.01$ and 1000 nodes.

3.2 Graph Models

The *Erdős-Rényi* or *ER* model was the first [8] *random graph* model. It is defined as follows: starting with N unconnected nodes, every pair of nodes is connected with probability p. Consequently, the total number of edges is a random variable with expectation value $E(n) = p\frac{N(N-1)}{2}$. A large number of interesting properties have been proven for this model, of which the following are important with respect to using it to embed a DAIS:

- If $p \geq \ln(N)/N$ the graph is totally connected.
- These graphs have the small world property. Their diameter scales logarithmically with the number of nodes and is concentrated around

$$d = \frac{\ln(N)}{\ln(pN)}$$

- The degree distribution follows a binomial distribution with parameters $N - 1$ and p. As figure 1 (dashed) shows, this means that the degrees of the nodes are concentrated around pN, resulting in a rather homogeneous graph.

The *Barabási-Albert* (*BA*) or *Scale-Free* model was first used in [2] to describe a wide variety of real-world networks that are all scale-free. That is, their degree distribution follows a power-law for large k : $P(k) \sim k^{-\gamma}$. Two ingredients are needed in the construction of such graphs :

- *Growth* Every node that is added to the graph links to m different nodes already present.
- *Preferential Attachment* The probability that a new node will link to node i depends on the degree k_i of node i as

$$\Pi(k_i) = \frac{k_i}{\sum_j k_j}$$

Following this procedure results in a graph with degree distribution $P(k) \sim k^{-3}$ for large k. Per definition, this kind of graph is connected. The number of edges is always equal to $2mN$ and it also has the small world property, the average path length increases approximately logarithmically with N. In contrast to the Erdős-Rényi model, the degree distribution always gives rise to a few nodes with very high degree (as e.g. in figure 1, solid line at $k = 32$). This makes the graph more heterogeneous, with a large majority of nodes having few connections but also some very highly connected ones. These nodes play an important role in the topology of the graph, as will become clear in section 4.

3.3 General Graph DAIS

To embed a DAIS in the above models, we need to derive the antibody migration behaviour. But in contrast to the ring and connected ones, the topologies of these models are not so trivial. The degree of their nodes can vary widely, having a large impact on the amount of communication between a node and it's neighbours. Nodes with more connections will receive more antibodies than ones with fewer connections and will sooner reach the number of antibodies necessary for the desired detection probability.

Therefore it makes sense to look at nodes with different degrees separately. A useful approximation to make is to regard a neighbouring node as totally average, with average number of antibodies N_a and average degree $\langle k \rangle$. Therefore, in a node with degree k, the number of antibodies entering at every time-step is equal to

$$in_k = \frac{N_a p_{am}}{\langle k \rangle} k \tag{9}$$

If we call $DN_a(k)$ the number of antibodies present at a node of degree k at time t, the amount of antibodies leaving each time-step is per definition $DN_a(k)p_{am}$.

The dynamics of this kind of system will converge to a state where the number of entering and leaving antibodies equals out. This happens when

$$DN_a(k)p_{am} = \frac{N_a p_{am}}{\langle k \rangle}\ k$$

So a node with degree k will converge to a state where it's number of antibodies is centred around

$$DN_a(k) = \frac{N_a}{\langle k \rangle}\ k \tag{10}$$

Figure 2 shows an illustration of this for a BA graph of 1000 nodes with $m = 5$. The figure shows plots of the average number of antibodies present in the nodes of degree 5 up to 15. All nodes initially have 100 antibodies. Depending on their degree they converge to a state where they have approximately $DN_a(k)$ antibodies.

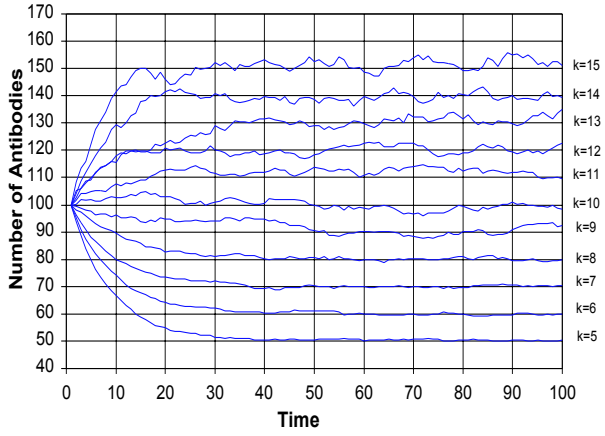


Figure 2: Average number of antibodies in nodes of degree k over time. In a BA graph of 1000 nodes with $m = 5$, $p_{am} = 0.1$, $N_a = 100$, $\langle k \rangle = 9.97$

So, for all nodes of degree k, at every time-step the chance that a random antibody will migrate to it is equal to

$$p_n(k) = \frac{in_k}{N_a N_n} = \frac{\frac{N_a p_{am}}{\langle k \rangle}\ k}{N_a N_n}$$

As before, the probability that an antibody has visited a node a number of times at time-step t follows the binomial distribution with parameters p_n and t. So, at time-step t an antibody has been in a node with degree k once or more times with probability

$$p_n(k)(t) = \sum_{i=1}^{t} C_i^t p_n(k)^i (1 - p_n(k))^{t-i} \tag{11}$$

Taking the weighted sum over the degree distribution of the graph, gives the final approximation of the antibody migration behaviour in an arbitrary graph

$$N_a(t) = N_a[1 + (N_n - 1)\sum_{i=1}^{N_n} P(i)p_n(i)(t)] \tag{12}$$

4 Error and Attack Tolerance

In the previous section we already noted that the ER and BA graph models contain far fewer edges (and hence consume less resources) than the fully-connected case, respectively $p\frac{N(N-1)}{2}$ and $2mN$ edges. In this section we use results from graph theory to show that, despite their relatively few edges, a DAIS embedded in these graphs can survive under high error rates and attacks. When nodes are removed from the graph, the performance will degrade gracefully until a critical threshold is reached and the graph suddenly collapses.

In the following, two types of node removal are looked at. *Error tolerance* is simulated by removing randomly selected nodes from the graph, corresponding to the random failure of nodes. In *Attack tolerance* the most connected nodes are removed, simulating a pre-determined attack with as much disruptive effect is possible.

4.1 Largest Cluster Size

Looking at the general antibody migration equation 12, it is clear that there are 2 values that determine how well a DAIS survives under error or attack conditions : the number of nodes N_n and $p_n(k)$. We will concentrate on the N_n factor. If the graph breaks up into 2 or more parts, or *clusters*, the set of nodes with which a node in the DAIS can communicate are suddenly limited to the ones in it's cluster. Since $N_a(t)$ scales linearly with the number of nodes (N_n), this greatly affects the detection performance.

The relative size of the largest cluster (i.e. the fraction of all nodes contained in the largest cluster) is an indicator of the state of fragmentation of the graph [1]. Figures 3 and 4 show plots of this measure for the ER and BA models, under both the error (solid lines) and attack (dashed lines) conditions. The ideal shape for the plots is the diagonal line, since it corresponds to the situation where the largest cluster contains all nodes that are still left in the graph.

4.2 Error Tolerance

Cohen *et al.* [5] and Callaway *et al.* [4] first studied the fragmentation of random and scale-free graphs under

random node failures. For both models, they conclude that there exists a *critical threshold* f_c for fragmentation at which the graph breaks down into tiny clusters. For ER graphs

$$f_c = 1 - \frac{1}{pN}$$

indicating that the larger the original average degree of the network, the larger the damage it can survive.

For Scale Free graphs with $\gamma > 3$

$$f_c = 1 - \frac{1}{\frac{\gamma-2}{\gamma-3}m - 1}$$

If $\gamma \leq 3$ however (as in the BA model used here, or the Internet), this critical threshold does not exist for infinite systems. For finite systems, a critical threshold is presents although at a very high node removal fraction > 0.99.

Figures 3 and 4 (solid lines), show a numerical simulation for a ER and BA graph under random node removal. As predicted, at $1 - pN = 1 - (0.006 \,.\, 500) = 0.7$ the ER graph starts to fragment into small clusters, while the BA graph shows no sudden fragmentation threshold. In summary, both models exhibit a very high degree of error tolerance, degrading gracefully up until a very high degree of node failure.

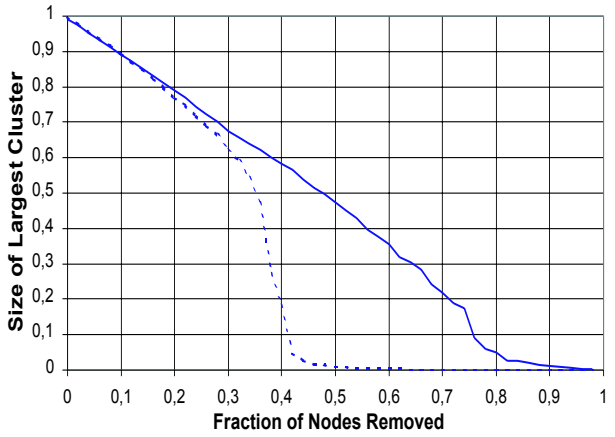


Figure 3: Relative Size of Largest Cluster in a ER graph with $N = 500$, $p = 0.006$ and hence $\langle k \rangle \approx 3$ under Error (solid) and Attack (dashed) conditions.

4.3 Attack Tolerance

As for attack tolerance, the results reached in the condensed matter physics community are not so clear yet. For both graph models, a sudden breakdown in the graph occurs when a critical fraction of nodes are removed. The figures illustrate this clearly. The BA

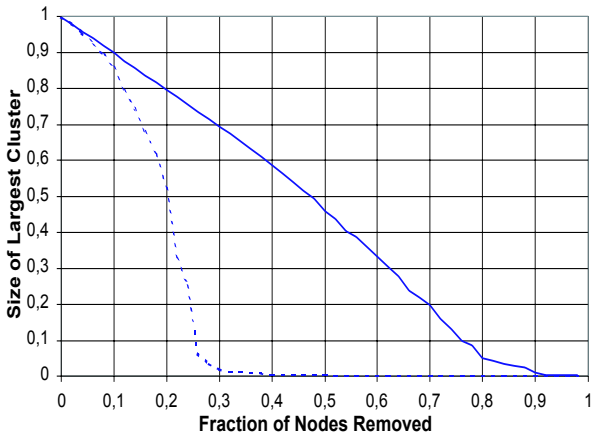


Figure 4: Relative Size of Largest Cluster in a BA graph with $N = 500$, and $\langle k \rangle \approx 3$ under Error (solid) and Attack (dashed) conditions.

model is much more sensitive to this kind of node removal because it contains a few, very important nodes with a very high degree that play an important role in the graph's topology. When these are progressively removed, the graph soon collapses. One recent result [5] concludes that for scale-free graphs with $\gamma > 2$ such a critical fraction exists. Since the ER model's critical threshold is higher than the BA model's, it should be used if tolerance against attacks is desired.

5 Possible Applications

As such, the above model and analysis can be applied to a large number of domains, not necessarily related to Artificial Immune Systems. In fact, all large scale communication networks containing a probabilistic element in the initiation of information exchanges can use it as a model. But here we will limit ourselves to some possible applications closely related to the original scope of this paper.

5.1 Applications

The first way of using it is as a plain parallelization of the standard AIS. The computational and memory complexity is distributed over the nodes in the graph, each taking on a part of the antibody generation and anomaly detection process. While the antibodies circulate, all nodes communicate back to one 'master' node which collects and synthesizes their result into one decision. More interesting is a truly distribution application. The nodes can be computers connected over a computer network (a LAN, or the Internet) running the same application (e.g. a Web Server). The

self-set consists of the normal data and/or instruction flow patterns of the application. These can be captured by monitoring the system-level calls or, when using Java technology, by hooking a custom-built code profiler into the Java Virtual Machine. The real difficulty here is in finding a suitable encoding for the large set of memory and instruction reference patterns that make up the normal behaviour of a large modern application. Alternatively, during the design or refactoring of a software product, this aspect can be taken into account and the program itself can be made to automatically output diagnostic behaviour to the DAIS. Other possibilities could be combining the ideas of this paper with the adaptive computer virus immune system introduced in [13] or the network intrusion detection models of [12] or [10]. Finally, applications where the robustness of the model is it's most important aspect can also be imagined. As shown above, given the right choice of graph, it degrades gracefully under failure or attack conditions.

5.2 Implementation

All simulations were generated using a Java implementation of the AIS and Graph Theory algorithms. The AIS package uses the standard bit-string representation or a generalized fuzzy set theory based one. For both representations, it includes the exponential time (from [9]) and greedy (from [6]) antibody generation algorithms, it can determine the optimal antibody morphology using an inductive algorithm and includes immune response models that use genetic antibody cloning. It has been tested on benchmark machine-learning datasets related to cancer diagnosis, the risk assessment of malaria cases and on a distributed physically security system using advanced video motion detection algorithms to encode the self-set. A visual front-end has been built that allows the user to connect to structured files or databases, visualize the data in 3D and acquire the self-set. All aspects of the AIS can be tuned via a GUI and it's behaviour can be monitored on new incoming data. To generate the results from the figures, several similar applications were linked together using the integrated support for the DAIS model described here.

6 Conclusion and Future Directions

6.1 Conclusion

We have shown how to construct large scale robust networks of communicating Artificial Immune Systems. A new model for a Distributed Artificial Immune Systems was introduced. We analysed the flow of antibodies in this DAIS model for the ring-shaped, fully connected and arbitrary graph topologies in order to arrive at a prediction for the detection performance over time in the nodes. The advantages and disadvantages of these topologies with respect to resource consummation and node failure tolerance were discussed using results from graph- and complex network theory. The ring-shaped and fully connected graphs were not robust against node failures or consumed too many resources. The ER and BA graphs on the other hand behaved excellent under node failure conditions, degrading gracefully up until very high failure rates. The authors feel that the primary contribution of the research is that we have shown how to predict the detection performance in a DAIS using these types of topologies. This result makes it possible to build large scale robust networks of artificial immune systems.

6.2 Future Directions

The DAIS model as well as it's implementation are currently being extended on several fronts. Work is underway to decentralize the graph building process. The graph topology construction algorithm in it's current state uses one central node that knows the entire graph structure to incrementally build the graph. This is unrealistic for very large scale networks and dangerous because it introduces a single point of failure. Due to the small-world properties of the graphs, it is possible to embed in every node the ability to add new nodes or even a self-healing graph construction algorithm that restores it's original topology's properties (e.g. degree distribution) after nodes have been removed.

Another track aims to heighten the detection performance by extending the immune system metaphor. In the biological immune system, molecules of the *Major Histocompatibility Complex* play the role of *antigen presentation* agents [15]. They bind to antigens and present them at the cell surface to the immune system's T cells. Each individual has a different set of MHC types, resulting in a slightly different resilience against antigens. This introduces a positive *population level* effect on the overall immunity of a species. This effect could be incorporated in the DAIS model as suggested in [11] by installing a random permutation mask on the self strings. This reduces the amount of holes [6] in the antibody space, especially near self-strings, heightening the detection probability of the system.

Not only antibodies could be circulated throughout the DAIS. An immune response model or antibody lifecycle model [11] can be included in the nodes. It could e.g. use genetic operators to clone successful

antibodies. This way, information about anomalies is automatically and rapidly distributed throughout the entire system.

The authors are now in the process of integrating these ideas into the DAIS model and adapting the implementation accordingly.

7 Acknowledgments

This work has been supported by Data4s Future Technologies, Leuven, Belgium and Starlab Research, Brussels, Belgium. Johan Kaers would like to thank Pedro Rosas for the helpful suggestions about the manuscript. Richard Wheeler would like to thank Prof. Peter Ross and Prof. Donald Michie for their support.

References

[1] R. Albert and A.-L. Barabási, *Statistic Mechanics of Complex Networks*, in Reviews of Modern Physics 74, 47, 2002.

[2] A.-L. Barabási, R. Albert, *Emergence of Scaling in Random Networks*, Science 286, 509-511.

[3] G. Bianconi, A.-L. Barabási, *Bose-Einstein condensation in complex networks*, e-print arXiv:cond-mat/011224 13 November 2000

[4] D. S. Callaway, M. E. J. Newman, S. H. Strogatz and D. J. Watts, *Network robustness and fragility : Percolation on random graphs*, e-print arXiv:cond-mat/0007300v2 19 October 2000

[5] R. Cohen, K. Erez, D. ben-Avraham, S. Havlin, *Resilience of the Internet to random breakdowns*, e-print arXiv:cond-mat/0007048v2 19 October 2000

[6] P. D'haeseleer, S. Forrest, P. Helman. *An immunological approach to change detection : Algorithms, analysis and implications*, in Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy, pages 110-119, IEEE Computer Society Press, Piscataway, New Jersey.

[7] P. D'haeseleer, *Further efficient algorithms for generating antibody strings*, Technical Report CS95-3, The University of New Mexico, Albuquerque, NM, 1995.

[8] P. Erdős, A. Rényi, 1959, Publ. Math. Debrecen **6**, 290.

[9] S. Forrest, A. S. Perelson, L. Allen, R. Cherukuri, *Self-nonself discrimination in a computer.*, in Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, Los Alamitos, CA : IEEE Computing Society Press, 1994.

[10] S.A. Hofmeyr, *An Immunological Model of Distributed Detection and its Application to Computer Security*, Ph.D. Thesis, University of New Mexico, May 1999

[11] S. A. Hofmeyr, S. Forrest, *Architecture for an Artificial Immune System*, in Evolutionary Compation 8(4): 443-473, 2000

[12] J. Kim, P. Bentley, *An Artificial Immune Model for Network Intrusion Detection*, Proceedings of the 7th European Congress on Intelligent Techniques - Soft Computing (EUFIT'99). Aachen, Germany. September 13-19, 1999.

[13] Gary B. Lamont, R. E. Marmelstein, D. A. Van Veldhuizen, *A Distributed Architecture for a Self-Adaptive Computer Virus Immune System*, Chapter 11 in New Ideas in Optimization. Ed., Corne, Dorigo, and Glover, McGraw Hill, 1999

[14] M. E. J. Newman, *Ego-Centered networks and the ripple effect -or- Why all your friends are weird*, e-print arXiv:cond-mat/0111070v1, 5 November 2001

[15] P. Parham, *The Immune System*, Garland Publishing/Elsevier Science, 2000

[16] R. Pastor-Satorras, A. Vespignani, *Optimal Immunisation of Complex Networks*, e-print arXiv:cond-mat/0107267, 30 July 2001

[17] S. H. Strogatz, *Exploring Complex Networks*, Nature, Vol 410, p268-276, Macmillan Magazines Ltd., 8 March 2001

Information Immune Systems

Dennis L. Chao and Stephanie Forrest
Department of Computer Science
University of New Mexico
Albuquerque, NM 87131 USA
{dlchao,forrest}@cs.unm.edu

Abstract

Many people are exposed to more information than they can process effectively. We describe an approach to building an *information immune system* that eliminates undesirable information before it reaches the user. This approach is inspired by natural immune systems that protect us from pathogens. The potential applications of an information immune system include filtering out undesirable data, generating a variety of solutions to a design problem, and finding consensus solutions to problems.

1 Introduction

Information overload is inevitable in a world that produces over an exabyte (one billion gigabytes) of information per year [28]. We will continue to produce and consume more and more information, so we must find innovative ways to manage it. Although finding and managing information are active research areas, much of the effort is directed towards active strategies such as information retrieval. Although these techniques help individuals locate desirable information, they also accelerate the information glut.

In this paper, we outline the features of an *information immune system* (IIS)[1] that could help people deal with the glut of data. We draw inspiration from natural immune systems that protect us from a seemingly limitless number of possible invaders such as bacteria, viruses, and parasites. We believe that an IIS can be constructed to eliminate undesired information after detecting it in a manner analogous to the natural immune system's. Such an IIS would be situated between an individual and a stream of information as a mediator. Instead of actively bringing more pieces of information to our attention, it will quietly censor unwanted data.

[1]The term "information immune system" was introduced by Neil Postman (in [39] and [40, page 63]).

An IIS should be capable of learning what kinds of information a user wants and discarding the rest. The task of distinguishing what is desirable is a difficult one. We propose taking one of the approaches used by our natural immune systems, which can "remember" a pathogen that infects us so it can eliminate it more quickly in future encounters. An IIS can do this by storing examples of rejected information and censoring similar data. If the memory of the system is too specific, this approach is likely to be ineffective. Pathogens and information can mutate over time, and our immune systems must be able to generalize. Therefore, both the natural and the information immune systems must also learn to eliminate *related* pathogens while taking care not to harm anything else.

An extension to a personal IIS is a group IIS. If one uses the IISs of many individuals in serial to filter a stream of information, the only information that can survive all IISs is the information that everyone finds desirable. We call such information "consensus solutions." Consensus solutions are useful in shared environments, such as broadcast music or artistic displays in public spaces. We will outline the relationship between a proposed IIS and a natural immune system, propose some applications of an IIS, including information filtering, interactive design, and collaborative design, then summarize the results of an experiment testing an IIS implementation.

2 Related work

Several areas of research have influenced our conception of an IIS. An IIS must be able to learn from past encounters, and the issues of learning and memory have long been addressed by the fields of artificial intelligence and machine learning. The primary task that we propose for an IIS, information filtering, has been explored by the field of human-computer interaction. Collaborative filtering may be relevant for IISs that classify data that are difficult to evaluate algorithmically. A few collaborative filtering systems make recommendations to groups instead of individuals. These group recommender systems perform a function similar to a group IIS. Finally, an IIS should be informed by

earlier work in artificial immune systems. All of these influences are briefly discussed below.

Case-based reasoning is a technique that adapts solutions to past problems to solve similar current problems [44]. Memory-based reasoning [49] and instance-based learning [1] are related schemes that use the solution of the most similar previous problem. Systems using these approaches learn by "remembering" specific past events rather than creating rules or generalizations. Immune memory uses a form of instance-based learning; the particular response that was effective in clearing a pathogen will likely be used in future encounters with related pathogens [29, 43, 5].

Associative memories, often called content-addressable memories, are neurally inspired architectures that can retrieve items using approximate addresses. Smith outlines the parallels between Kanerva's sparse distributed memory [25] and the memory of the natural immune system [47]. The memory of the natural immune system is not exact, and exposure to a novel pathogen can elicit the response primed by a related pathogen.

The term "information filtering" refers to a large range of techniques used to remove data from an incoming stream on the basis of user- or group- specified preferences [2]. Early approaches used simple rules [30] or signatures (e.g. keywords) to identify undesirable data to block. These approaches are still popular, and many commercial products, such as Cyberpatrol [9] for web content and the Realtime Blackhole List [41] and Brightmail [6] for e-mail, come with long lists of rules and signatures, which can be effective in blocking undesirable data but are vulnerable to malicious sources that can craft information to bypass them. To thwart these adaptive adversaries and to personalize the filtering, the user is often allowed to specify additional rules for accepting and rejecting data. Unfortunately, the specification of such rules is often difficult and error-prone, and therefore not used routinely. An IIS should incorporate reliable signatures of undesirable data as a first line of defense to be supplemented with more adaptive techniques to provide better and more personalized coverage.

Several research systems simplify the filter specification problem by placing the burden of generating rules on software rather than on a user or programmer. Infoscope [14] monitors a user's behavior to create rules for Usenet newsgroup filtering. The system suggests these rules to the user, who can accept, modify, or reject them. Maxims [34], an interface agent for e-mail, also generates filtering rules based on user behavior, but it suggests actions for the user to take rather than rules when it is confident in its predictions. Rule-based learning schemes often require many examples before they can infer new rules. In contrast, an IIS using an instance-based learning approach could learn to block a class of data upon seeing only a single exemplar.

Collaborative filtering uses the preferences of others to help an individual make choices [31, 16, 42]. For example, a collaborative filtering system would recommend an item for a person to purchase by choosing an item purchased by someone with a similar purchase history. By harnessing the collective preferences of many individuals, such systems can infer similarity between items without needing to understand the relationship between them. This approach is useful when it is difficult for a program to determine similarities between items, such as art or music. An IIS could incorporate collaborative filtering techniques to determine the similarity between items for its associative memory capabilities.

There are a few systems that recommend items to groups instead of individuals. MusicFX [32] selects music stations that are broadcast to a gym full of people. The members of the gym must rate all the stations beforehand, and MusicFX plays one of the stations with the highest average rating. One shortcoming of MusicFX is that it apparently does not scale to a large number of choices. If the users were not able to evaluate all of the stations, the quality of the system's choices would likely be degraded. GroupCast [33], developed by the same research group, used a conceptually similar scheme to display content on a public display system. Unfortunately, they found that the necessary user profiles would have been too large for any user with a reasonable amount of patience to complete. In addition, without extensive profiles it was difficult to find appropriate intersections of user preferences to put on the GroupCast displays. Instead, they displayed content that was interesting to *one* of the users, hoping that by chance others would have similar interests. PolyLens [36] recommends movies to small groups of people who watch movies together. This system applies a standard collaborative filtering algorithm to make recommendations for each of the group members then combines the results to make a group recommendation. These systems give insight into the nature of finding solutions for groups. Notably, it is difficult to make recommendations that satisfy all members of a large group.

Immune system inspired algorithms have often been used for anomaly detection. They draw on the metaphor of the adaptive immune system's ability to distinguish between *self*, or normal data, and *nonself*, or anomalous data. One of the first such systems was the negative selection algorithm introduced by Forrest *et al* [15]. The algorithm generated random strings and those that were similar to sequences of bytes in a given computer file were eliminated. The surviving strings were therefore not similar to any in the file. If one of these strings ever matched the contents of the file, then this indicated that the contents had been changed since the training period. These strings were used as *negative detectors* to detect novel sequences of bytes, such as those introduced when a virus corrupts or infects a file. The ARTIS framework is an extension of this work

that applies negative selection to detect anomalies in streams of data rather than in static data sets [18, 19]. This framework was used to create systems to detect network intrusions [18, 19, 27, 54].

We believe that most useful sources of information present continually changing streams of data, so that it would be undesirable for an IIS to reject all novel data. The IIS is inspired by the immune system's ability to remember past encounters with pathogens, while the artificial immune system approach to anomaly detection is usually based on the immune system's ability to detect novel foreign proteins. The anomaly detection ability of ARTIS could complement an IIS for certain applications, but for many applications we imagine using negative detectors without negative selection.

Many computer scientists have developed artificial immune systems based on idiotypic network theory [24]. The idiotypic systems focus on the dynamics of the interactions among similar antibodies and antigens. Although many do not attempt to reproduce the behaviors seen in the natural immune system, they have useful properties that have been applied to search [4], data classification [21], cluster detection [50], and data mining [12]. The classifications produced by idiotypic artificial immune systems could potentially be used as metadata to enhance the discrimination of an IIS.

3 The immune system as an information filter

We believe that an IIS can borrow several pattern recognition mechanisms from the natural immune system. Our natural immune system consists of two components that use different pathogen recognition strategies. The *innate* immune system uses a few reliable signatures of foreignness to identify invaders, which Janeway calls pathogen-associated molecular patterns (PAMP) [22]. An example of a PAMP is the mannose carbohydrate molecules found on many bacteria and other pathogens but not in mammals [48]. These signatures have been stable over evolutionary time and are encoded in the genome of our immune systems. This strategy is used by many of the signature and rule based information filtering products mentioned in Section 2. These products could serve as a first line of defense, playing the role of the innate immune system in an IIS. However, not all signatures of pathogens have been (or even can be) anticipated, and evolution will favor pathogens that do not carry the signatures recognized by our innate immune systems. One role of the *adaptive* immune system, discussed below and outlined in Table 1, is to discover the signatures of pathogens not covered by the innate immune system. In the following subsections we describe some issues that an IIS must face and how one can draw inspiration from the natural immune system to address them.

Natural IS	Information IS
shape space	parameter space
self	desirable information
non-self	undesirable information
helper T cell	user's judgment
costimulation	rejection of information by user
naïve cells	implicit (not instantiated)
active lymphocyte	detector
memory lymphocyte	detector
cytolytic activity	censor solution
cross-reactive radius	detector radius
thymic selection	protecting known desirable information
illness	user exposed to undesirable data

Table 1: The immunological analogy made explicit.

3.1 Negative detectors and shape space

An IIS should be able to remember which pieces of information a user rejected in the past so it can censor them in the future. However, the strategy of rejecting each item individually is ineffective when one is faced with a seemingly limitless variety of information. An IIS must be able to generalize; rejecting one item should implicitly reject similar items. The natural immune system has this ability.

The adaptive immune system has a repertoire of lymphocytes that detect pathogens. Each lymphocyte is specific to a particular antigen, or protein signature, expressed by pathogens. If a lymphocyte detects a cell with a matching signature, it may destroy it. However, pathogens may mutate and subtly change their antigenic profiles, so lymphocytes should also be able to recognize close variants. Perelson and Oster suggested the conceptual framework of *shape space* [38], a high-dimensional space that represents the universe of possible antigens. Every antigen has a location in shape space, and small mutations in a pathogen may alter its proteins, thus shifting its location in shape space. For a lymphocyte to be effective, it should be able to cover a large enough area in shape space that most mutations would not evade detection. The area in shape space that a lymphocyte covers is sometimes known as its *ball of stimulation* because it is postulated that a lymphocyte can recognize an antigen within a certain radius of its location in shape space.

An IIS could use negative detectors to censor information that the user does not want. As with the natural immune system, a detector should be able to cover a volume in shape space, not just a point. Therefore, it is necessary for an IIS to have some notion of the similarity between two pieces of information. Two items that are similar are close in "information space." Collaborative filtering techniques could be used in cases in which it is too difficult to define a function that en-

codes the subjective similarity between two pieces of information.

3.2 Costimulation

Because everyone has different informational needs, each IIS user should be able to decide which types of data to reject. Many information filters require the user to write rules to customize the filtering, but we believe that the user should need only to identify exemplars of undesirable information. Once the user rejects a piece of information, an IIS should be able to automatically reject similar information in the future.

In the adaptive immune system, helper T cells are generally required to *costimulate*, or activate, cells in the presence of a novel pathogen. Helper T cells provide confirmation that a pathogen should be eliminated. This process reduces the chances of immune cells attacking the body, which is known as an autoimmune response. Once costimulated, the effector cell becomes active and can attempt to eliminate the invader, whether by releasing antibodies in the case of B cells or by killing the infected cells directly in the case of cytotoxic T cells. Some co-stimulated cells become memory cells, which are long-lived. In future encounters with the same pathogen, memory cells have lesser or even no costimulation requirement.

In an IIS, the user could adopt the role of the helper T cells by providing costimulation signals to the system, an idea introduced in [19]. The idle cells waiting for costimulation are implicit—only detectors corresponding to active or memory cells need to be instantiated. When the user rejects a piece of information, a detector specific to that item would be created. These detectors would prevent any similar data from being presented in the future. The user's only responsibility would be to inform the IIS when undesirable data are being presented.

3.3 The addition of negative selection

When the user has rejected a sufficient amount of information, the space not covered by detectors approximates the space of useful information (Figure 1). Unfortunately, useful information that is too similar to unwanted information runs the risk of being censored by an IIS negative detector. Therefore, we suggest incorporating a technique that the adaptive immune system uses to prevent the immune system from attacking the body's own cells.

The adaptive immune system uses thymic selection to eliminate T cells that may harm the body. Before T cells can enter the repertoire, they are exposed to a large sample of the body's own proteins. Those that bind too tightly to one of the body's proteins are eliminated in a process known as negative selection. Therefore, the T cells that survive are not likely to recognize a self protein.

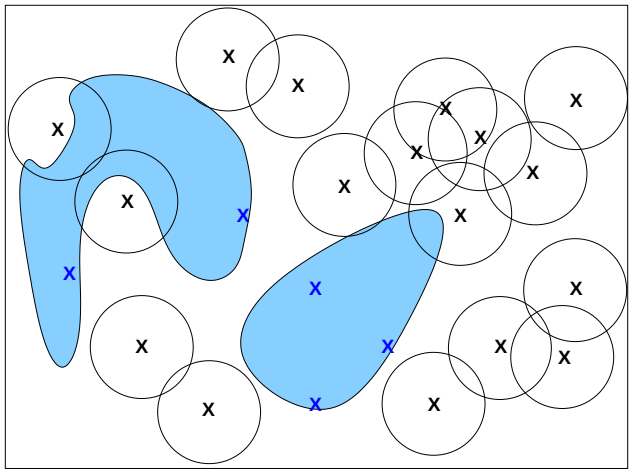


Figure 1: Coverage in the model without negative selection. The shaded regions represent information that is useful. The circles represent the extent of active detector coverage. The Xs without circles represent the detectors that should never be costimulated because they are within the regions of acceptable solutions.

A similar strategy could be employed in advance by an IIS to protect types of information known to be useful. These types could be declared "off-limits" to the IIS and would be allowed to bypass the IIS to reach the user. This is especially useful when the characteristics of certain desirable information are known *a priori*. For example, the IISs of a company's employees should probably not be allowed to eliminate official company e-mail. When a user costimulates a solution whose detector would cover some desirable information, the system could ignore the costimulation signal because there should be no "implicit" detectors in this region. No information from the "good" regions of information space will ever be censored by the detectors (Figure 2).

3.4 The role of senescence

Users may want to filter out some types of information for only a short period of time. For example, if a radio station plays a song too frequently or if a news story receives too much coverage, a listener may tire of it. These individuals may actually enjoy hearing the song or listening to new developments in the news story at a later date, so the detectors would be counterproductive after their "natural" lifetimes.

Active immune cells have short lifetimes, and memory cells can be eliminated by competition for space [45]. These features may be desirable in the algorithm for two reasons. The first is to provide "rolling coverage" of self. If the fitness function (e.g. the user's tastes) change over time, one could have the lifetime of the active immune cells be finite to reflect the dynamic nature of the user's judgment. The second reason is space efficiency. It may not be feasible to store an unbounded number of detectors. One could "age out" old detec-

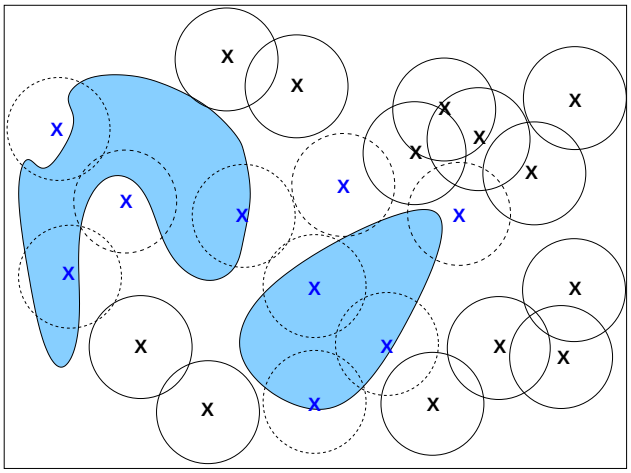


Figure 2: Coverage in the model using negative selection. The dotted circles represent the extent of detectors that are eliminated by negative selection. The solid circles are regions covered by active detectors. Note that none of the useful information protected by negative selection (the shaded regions) can be covered by detectors.

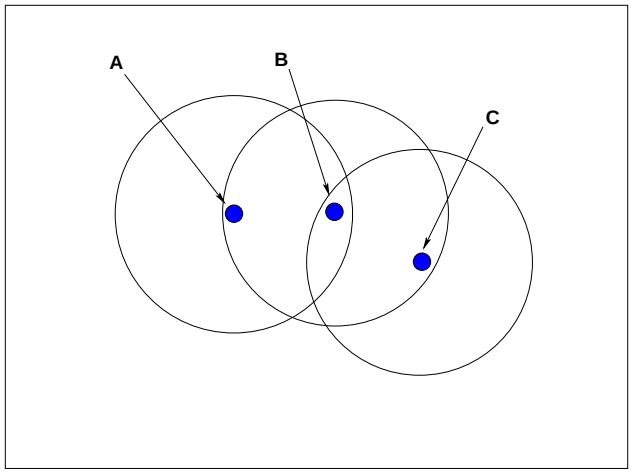


Figure 3: The effect of history. The dots labeled "A" and "B" and "C" represent solutions the user does not like. The circles are the extents of their negative detectors, or "balls of stimulation". If solution A is rejected by the user first, the detector that forms around it would reject B before it could be presented. However, C could be presented because it does not fall within the scope of the detector for A. However, if B had been presented first, the story would be different. Neither A nor C would be seen after B because its detector would cover all three solutions.

tors to make room for new ones. Alternatively, the user could manually create memory detectors to cover patterns that he or she never wants to see again.

3.5 The effect of history

The order in which an IIS is exposed to information can have impact on its effectiveness. Such phenomena have been observed in the natural immune system, particularly in the case of influenza. Immunologists have discovered that the response to a strain of flu may be dominated by cells that were created in response to an earlier exposure to a *different* strain [10, 13]. These memory cells are probably most effective against the strain that generated them, but they can respond to related ones. This phenomenon is known as original antigenic sin, and many vaccines take advantage of this effect. For example, if one is exposed to the relatively harmless cowpox bacteria, one is protected against the related but deadly smallpox [23]. Unfortunately, prior exposure to antigens can also work against us [46]. For example, a flu vaccine works by eliciting a mild response to a particular strain's flu antigens so that an individual will be able to mount an effective secondary response when exposed to it in the future. However, the memory cells created by a vaccine from a previous year may attack and eliminate subsequent vaccines before they can establish protective immunity. If the first vaccine does not provide protection against the strains corresponding to these later vaccinations, this individual would be vulnerable to them (Figure 3). If this individual had not received this first vaccine, the subsequent vaccines could have been effective.

One should be able to "vaccinate" an IIS by exposing it to undesirable information without necessarily exposing the user. This would allow an administrator to preemptively block the passage of certain kinds of information to a user. For example, a corporation might prohibit certain kinds of e-mail or web traffic, such as pornography or personal e-mail. The corporation could "vaccinate" the IISs of its employees with exemplars from the banned categories, and the employees would not be exposed to these kinds of information. Because the order in which an individual is exposed to undesirable information may affect the coverage of the individual's IIS, the vaccination strategy should be planned with care.

4 Applications

The most obvious use of an IIS of the sort described here is information filtering. An IIS could serve as a personalized interface agent that learns a user's preferences for sources of information or for a range of options that is too large or dynamic for a user to evaluate. Because it only requires feedback when the user is exposed to something he or she does not want and it learns without using separated training and testing phases, an IIS could be a non-intrusive addition to many user interfaces. It could complement active strategies, such as information retrieval, that search for potentially useful information.

The IISs of individuals can be combined to produce a group IIS. One can think of an IIS as a sieve that fil-

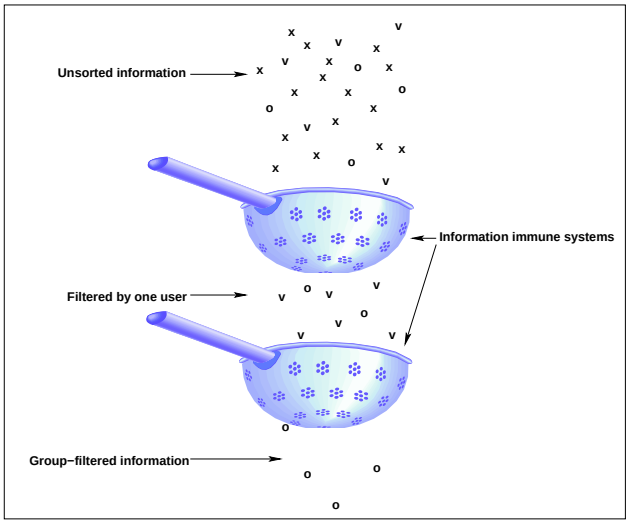


Figure 4: An information immune system as a sieve. The IIS stands between a stream of information and a user, blocking a significant portion of it. Only the information that can pass through the "sieve" actually reaches the user. When the IISs of multiple users are applied in serial, the information that passes all IISs are "consensus solutions".

ters undesirable information. The data that can pass through the "sieves" of many people are those that are likely to satisfy all of them (Figure 4). We call these data *consensus solutions*. A group IIS would be useful when the group is exposed to shared information. For example, if co-located people want to listen to music together, one would want to play music that none of the individuals dislikes. It remains to be seen how well a group IIS will work with a large group of users in a particular domain. Consensus-finding will become more important with the increase in the number of intelligent environments that automatically respond to the users' needs. For example, smart home technology can adjust the music, artwork, temperature, and lighting to accommodate its occupants. Most research focuses on catering to a single individual [26, 17, 51, 20], but for many environments it will be important to satisfy the preferences of multiple occupants.

An IIS could be used to assist designers and artists [7]. If a random source of design solutions or works of art were fed to an IIS, only those that are not similar to those rejected in the past would pass through. The quality of the solutions from this filtered stream should be significantly better than the unfiltered stream. This could be a useful strategy for design problems in which a designer or artist is interested in exploring a large range of possible solutions. The solutions could be refined or optimized using other techniques, such as evolutionary design [3].

Collaborative design could be facilitated by using the IISs of multiple individuals. The combination of IISs is the superset of solutions that people dislike. Consensus solutions are not optimal solutions, but a variety of solutions that are "good enough" for *everyone*. In certain cases, it would be preferable to combine the favored solutions of each of the group members instead of using a group IIS. This could be done by taking the intersection of the favored solutions of the members or by combining (hybridizing) them. The former strategy is problematic when the solution space is too large for a user to specify the set of all acceptable solutions (as was found with GroupCast[33]) or if the knowledge of a user's preferences is incomplete. In these cases, intersections will be difficult to find. The latter strategy of combining solutions can be difficult. It is often not obvious how to combine the desirable traits of two solutions to produce a third good solution. By combining the *dislikes* of multiple users, the space of potential candidate solutions is likely to be larger and there is no need to combine solutions.

5 An example: An aesthetic information immune system

We have applied the principles discussed in this paper to design a simple IIS that generates computer art [7], and we summarize the results here. The IIS characterized several users' preferences for a particular family of computer-generated images known as Biomorphs [11]. Biomorphs are recursively drawn figures that can be defined by nine parameters. Each user was shown a set of randomly generated Biomorphs and instructed to reject those that he or she did not like. For each user, an IIS was created based on the parameters of the rejected Biomorphs. The IIS filtered out any images that had parameters similar to those rejected in the past, and they formed a rough estimate of the parts of Biomorph parameter space that each user wanted to avoid.

We tested whether a user could use an IIS to filter a stream of randomly generated Biomorphs to produce an edited stream of high quality Biomorphs, based on the subjective judgments of the user. We also investigated group IISs that applied the IISs of several users in serial. We wanted to determine if the addition of other users' IISs would enhance or degrade the quality of a single user's IIS. These effects were measured by having the users evaluate three sets of randomly generated Biomorphs that were filtered using no IIS, their own IIS, a group IIS composed of seven users' IISs. Most users preferred the Biomorph images filtered using their own IISs to the unfiltered ones, suggesting that the IISs had preferentially filtered out images that would have been rejected by the users. The group IIS was less successful, possibly because of differences among the users' Biomorph aesthetic preferences or possibly because of the coarseness of the detectors (we used quite coarse-grained detectors in order to reduce the training time for each user). We repeated the test with a subset of three users and a group IIS

composed of only these three users' IISs. The images produced by this smaller group's IIS were perceived to be better than unfiltered, and each user found these images to be no worse than those produced using their own IISs, indicating the possibility of a consensus solution.

6 Conclusions

We believe that information immune systems could play an important role in this age of information overload. To date, we as a society have developed only crude coping mechanisms to allow us to survive the enormous amounts of data to which we are routinely exposed [35]. A successful IIS would reduce the load and make other strategies for finding and processing information more effective.

Information immune systems, however, should be fielded with caution. As filtering strategies become more sophisticated, the producers of unwanted information will themselves adapt, creating a kind of information arms race. We see this already in the adaptation of magazine advertisements designed to resemble content articles and "junk mail" packaged in official-looking envelopes. Even more insidious techniques embed advertising in content in which people are interested. Advertisements can be wrapped around e-mail for presentation before the user can receive it [8], corporate logos and products can be digitally edited into films and television programs [53], and some shows integrate their sponsors' products into the plotlines [37]. Even in the absence of adaptive adversaries, our information filtering technology will drive a selective process that will minimize the differences between desirable and undesirable information. As our filters gain efficacy, undesirable information will evolve to evade them. The filters must constantly co-evolve or else they will rapidly become useless. When we begin deploying IISs, we must be prepared to live in a dynamic information ecosystem in which our defenses must adapt as quickly as the abilities of unwanted information to penetrate them [52].

Acknowledgments

We thank Marc Millier of Intel Corporation for suggesting the term "information immune system." The authors gratefully acknowledge the support of the National Science Foundation (ANIR-9986555), the Office of Naval Research (N00014-99-1-0417), Defense Advanced Projects Agency (AGR F30602-00-2-0584), the Intel Corporation, and the Santa Fe Institute.

References

[1] D. Aha, D. W. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992.

[3] P. J. Bentley, editor. *Evolutionary Design by Computers*. Morgan Kaufmann Publishers, San Francisco, California, 1999.

[4] H. Bersini and F. J. Varela. Hints for adaptive problem solving gleaned from immune networks. In H. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature*, pages 343–354, Berlin, 1991. Springer-Verlag.

[5] J. A. Borghans, A. J. Noest, and R. J. De Boer. How specific should immunological memory be? *J Immunol*, 163(2):569–75, 1999.

[6] *Brightmail Solution Suite*. Brightmail, Inc., San Francisco, California, 2002. http://www.brightmail.com.

[7] D. L. Chao and S. Forrest. Generating biomorphs with an aesthetic immune system. In *Artificial Life VIII: Proceedings of the Eighth International Conference on the Simulation and Synthesis of Living Systems*, 2002. (in press).

[8] N. Cochrane. Mobile entrepreneur rapt in wireless e-mail advertising. *The Age (Melbourne)*, 24 Jul 2001:7.

[9] *Cyberpatrol*. SurfControl plc, Westborough, Massachusetts, 2002.

[10] F. M. Davenport, A. V. Hennessy, and T. Francis. Epidemiologic and immunologic significance of age distribution to antibody to antigenic variants of influenza virus. *J Exp Med*, 98:641–656, 1953.

[11] R. Dawkins. *The Blind Watchmaker*. Longman Scientific and Technical, Harlow, UK, 1986.

[12] L. N. De Castro and F. J. Von Zuben. aiNet: An artificial immune network for data analysis. In H. A. Abbass, R. A. Sarker, and C. S. Newton, editors, *Data Mining: A heuristic approach*, chapter XII, pages 231–259. Idea Group Publishing, USA, Hershey, Pennsylvania, 2001.

[13] S. Fazekas de St. Groth and R. G. Webster. Disquisitions of original antigenic sin. I. Evidence in man. *J Exp Med*, 124(3):331–45, 1966.

[14] G. Fischer and C. Stevens. Information access in complex, poorly structured information spaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 1991)*, pages 63–70, 1991.

[15] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In *Proceedings of the 1994 IEEE*

Symposium on Research in Security and Privacy. IEEE Computer Society Press, 1994.

[16] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[17] S. R. Hedberg. After desktop computing: A progress report on smart environments research. *IEEE Intelligent Systems*, 15(5):7–9, 2000.

[18] S. A. Hofmeyr. *An immunological model of distributed detection and its application to computer security*. PhD thesis, University of New Mexico, Albuquerque, New Mexico, 1999.

[19] S. A. Hofmeyr and S. Forrest. Architecture for an artificial immune system. *Evolutionary Computation*, 7(1):1289–1296, 1999.

[20] *House_n Living Laboratory*. School of Architecture and Planning, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2001.

[21] J. E. Hunt and D. E. Cooke. Learning using an artificial immune system. *Journal of Network and Computer Applications*, 19:189–212, 1996.

[22] C. A. Janeway Jr. The immune system evolved to discriminate infectious nonself from noninfectious self. *Immunol Today*, 13(1):11–6, 1992.

[23] E. Jenner. *An Inquiry into the Causes and Effects of the Variolae Vaccinae; a Disease Discovered in some of the Western Counties of England, Particularly Gloucestershire, and Known by the Name of the Cow Pox*. 1798.

[24] N. K. Jerne. Towards a network theory of the immune system. *Ann Immunol (Inst Pasteur)*, 125C:373–389, 1974.

[25] P. Kanerva. *Sparse Distributed Memory*. MIT Press, 1988.

[26] C. D. Kidd, R. J. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa, B. MacIntyre, E. Mynatt, T. E. Starner, and W. Newstetter. The aware home: A living laboratory for ubiquitous computing research. In *Proceedings of the Second International Workshop on Cooperative Buildings (CoBuild'99)*, pages 191–198, 1999.

[27] J. Kim and P. J. Bentley. The artificial immune model for network intrusion detection. In *Proceedings of the 7th European Conference on Intelligent Techniques and Soft Computing (EUFIT'99)*, 1999.

[28] P. Lyman and H. R. Varian. How much information?, 2000. Retrieved from http://www.sims.berkeley.edu/how-much-info.

[29] C. R. Mackay, W. L. Marston, L. Dudler, O. Spertini, T. F. Tedder, and W. R. Hein. Tissue-specific migration pathways by phenotypically distinct subpopulations of memory T cells. *Eur J Immunol*, 22(4):887–95, 1992.

[30] T. W. Malone, K. R. Grant, and F. A. Turbak. The Information Lens: An intelligent system for information sharing in organizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 1986)*, pages 1–8, 1986.

[31] T. W. Malone, K. R. Grant, F. A. Turbak, S. A. Brobst, and M. D. Cohen. Intelligent information sharing systems. *Communications of the ACM*, 30(5):390–402, 1987.

[32] J. F. McCarthy and T. D. Anagnost. MusicFX: An arbiter of group preferences for computer supported collaborative workouts. In *Proceedings of the ACM 1998 Conference on Computer Supported Cooperative Work*, pages 363–372. ACM Press, 1998.

[33] J. F. McCarthy, T. J. Costa, and E. S. Liongosari. UniCast, OutCast & GroupCast: An exploration of new interaction paradigms for ubiquitous, peripheral displays. In *Workshop on Distributed and Disappearing User Interfaces in Ubiquitous Computing at the SIGCHI Conference on Human Factors in Computer Systems (CHI 2001)*. ACM Press, 2001.

[34] M. Metral. *A Generic Learning Interface Agent*. B.Sc. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1992.

[35] J. G. Miller. Information input overload and psychopathology. *American Journal of Psychiatry*, 116(8):695–704, 1960.

[36] M. O'Connor, D. Cosley, J. A. Konstan, and J. Riedl. PolyLens: A recommender system for groups of users. In *Proceedings of the 7th European Conference on Computer Supported Cooperative Work (ECSCW 2001)*, pages 199–218. Kluwer Academic, 2001.

[37] G. Pennington. Just try zapping these ads. *St. Louis Post-Dispatch*, 14 Apr 2002:F1.

[38] A. S. Perelson and G. F. Oster. Theoretical studies of clonal selection: minimal antibody repertoire size and reliability of self-non-self discrimination. *J Theor Biol*, 81(4):645–70, 1979.

[39] N. Postman. Informing ourselves to death, 11 Oct 90. Speech delivered to the German Informatics Society (Gesellschaft für Informatik).

[40] N. Postman. *Technopoly. The Surrender of Culture to Technology.* Vintage Books, New York, 1992.

[41] *Realtime Blackhole List.* Mail Abuse Prevention System LLC, Redwood City, California, 2002. http://www.mail-abuse.org/rbl/.

[42] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186. ACM Press, 1994.

[43] F. Sallusto, D. Lenig, R. Forster, M. Lipp, and A. Lanzavecchia. Two subsets of memory T lymphocytes with distinct homing potentials and effector functions. *Nature*, 401(6754):708–12, 1999.

[44] R. C. Schank. *Dynamic Memory: A theory of reminding and learning in computers and people.* Cambridge University Press, New York, 1982.

[45] L. K. Selin, K. Vergilis, R. M. Welsh, and S. R. Nahill. Reduction of otherwise remarkably stable virus-specific cytotoxic T lymphocyte memory by heterologous viral infections. *J Exp Med*, 183(6):2489–99, 1996.

[46] D. J. Smith, S. Forrest, D. H. Ackley, and A. S. Perelson. Variable efficacy of repeated annual influenza vaccination. *Proc Natl Acad Sci U S A*, 96(24):14001–6, 1999.

[47] D. J. Smith, S. Forrest, and A. S. Perelson. Immunological memory is associative. In *Workshop Notes, Workshop 4: Immunity Based Systems, Intnl. Conf. on Multiagent Systems*, pages 62–70, 1998.

[48] P. D. Stahl and R. A. Ezekowitz. The mannose receptor is a pattern recognition receptor involved in host defense. *Curr Opin Immunol*, 10(1):50–5, 1998.

[49] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, 1986.

[50] J. Timmis. *Artificial immune systems: A novel data analysis technique inspired by the immune network theory.* PhD thesis, University of Wales, 2000.

[51] The user in control. *Philips Research Password*, 3:10–13, 2000.

[52] L. Van Valen. A new evolutionary law. *Evolutionary Theory*, 1:1–30, 1973.

[53] Virtual ads, real problems. *Advertising Age*, 70(22):30, 1999.

[54] P. D. Williams, K. P. Anchor, J. L. Bebo, G. H. Gunsch, and G. D. Lamont. CDIS: Towards a computer immune system for detecting network intrusions. In W. Lee, L. Me, and A. Wespi, editors, *4th International Symposium, Recent Advances in Intrusion Detection*, pages 117–133, Berlin, 2001. Springer-Verlag.

The Danger Theory and Its Application to Artificial Immune Systems

Uwe Aickelin
Department of Computing
University of Bradford
Bradford
BD7 1DP
u.aickelin@bradford.ac.uk

Steve Cayzer
Hewlett-Packard Laboratories
Filton Road
Bristol
BS12 6QZ
Steve_Cayzer@hp.com

Abstract

Over the last decade, a new idea challenging the classical self-non-self viewpoint has become popular amongst immunologists. It is called the Danger Theory. In this conceptual paper, we look at this theory from the perspective of Artificial Immune System practitioners. An overview of the Danger Theory is presented with particular emphasis on analogies in the Artificial Immune Systems world. A number of potential application areas are then used to provide a framing for a critical assessment of the concept, and its relevance for Artificial Immune Systems.

1 INTRODUCTION

Over the last decade, a new theory has become popular amongst immunologists. It is called the Danger Theory, and its chief advocate is Matzinger [18], [19] and [20]. A number of advantages are claimed for this theory; not least that it provides a method of 'grounding' the immune response. The theory is not complete, and there are some doubts about how much it actually changes behaviour and / or structure. Nevertheless, the theory contains enough potentially interesting ideas to make it worth assessing its relevance to Artificial Immune Systems.

It should be noted that we do not intend to defend this theory, which is still controversial [21]. Rather we are interested in its merits for Artificial Immune System applications and hence its actual existence in the humoral immune system is of little importance to us. Our question is: Can it help us build better Artificial Immune Systems?

Few other Artificial Immune System practitioners are aware of the Danger Theory, notable exceptions being Burgess [5] and Willamson [22]. Hence, this is the first paper that deals directly with the Danger Theory, and it is the authors' intention that this paper stimulates discussion in our research community.

In the next section, we provide an overview of the Danger Theory, pointing out, where appropriate, some analogies in current Artificial Immune System models. We then assess the relevance of the theory for Artificial Immune System security applications, which is probably the most obvious application area for the danger model. Other Artificial Immune System application areas are also considered. Finally, we draw some preliminary conclusions about the potential of the Danger concept.

2 THE DANGER THEORY

The immune system is commonly thought to work at three levels: External barriers (skin, mucus), innate immunity and the acquired or adaptive immune system. As part of the third and most complex level, B-Lymphocytes secrete specific antibodies that recognise and react to stimuli. It is this pattern matching between antibodies and antigens that lies at the heart of most Artificial Immune System implementations. Another type of cell, the T (killer) lymphocyte, is also important in different types of immune reactions. Although not usually present in Artificial Immune System models, the behaviour of this cell is implicated in the Danger model and so it is included here. From the Artificial Immune System practitioner's point of view, the T killer cells match stimuli in much the same way as antibodies do.

However, it is not simply a question of matching in the humoral immune system. It is fundamental that only the 'correct' cells are matched as otherwise this could lead to a self-destructive autoimmune reaction. Classical immunology [12] stipulates that an immune response is triggered when the body encounters something non-self or foreign. It is not yet fully understood how this self-non-self discrimination is achieved, but many immunologists believe that the difference between them is learnt early in life. In particular it is thought that the maturation process plays an important role to achieve self-tolerance by eliminating those T and B cells that react to self. In addition, a 'confirmation' signal is required; that is, for either B cell or T (killer) cell activation, a T (helper) lymphocyte must also be activated. This dual activation is

further protection against the chance of accidentally reacting to self.

Matzinger's Danger Theory debates this point of view (for a good introduction, see Matzinger [18]). Technical overviews can be found in Matzinger [19] and Matzinger [20]. She points out that there must be discrimination happening that goes beyond the self-non-self distinction described above. For instance:

- There is no immune reaction to foreign bacteria in the gut or to the food we eat although both are foreign entities.
- Conversely, some auto-reactive processes are useful, for example against self molecules expressed by stressed cells.
- The definition of self is problematic – realistically, self is confined to the subset actually seen by the lymphocytes during maturation.
- The human body changes over its lifetime and thus self changes as well. Therefore, the question arises whether defences against non-self learned early in life might be autoreactive later.
- Other aspects that seem to be at odds with the traditional viewpoint are autoimmune diseases and certain types of tumours that are fought by the immune system (both attacks against self) and successful transplants (no attack against non-self).

Matzinger concludes that the immune system actually discriminates "some self from some non-self". She asserts that the Danger Theory introduces not just new labels, but a way of escaping the semantic difficulties with self and non-self, and thus provides grounding for the immune response. If we accept the Danger Theory as valid we can take care of 'non-self but harmless' and of 'self but harmful' invaders into our system. To see how this is possible, we will have to examine the theory in more detail.

The central idea in the Danger Theory is that the immune system does not respond to non-self but to danger. Thus, just like the self-non-self theories, it fundamentally supports the need for discrimination. However, it differs in the answer to what should be responded to. Instead of responding to foreignness, the immune system reacts to danger.

This theory is borne out of the observation that there is no need to attack everything that is foreign, something that seems to be supported by the counter examples above. In this theory, danger is measured by damage to cells indicated by distress signals that are sent out when cells die an unnatural death (cell stress or lytic cell death, as opposed to programmed cell death, or *apoptosis*).

Figure 1 depicts how we might picture an immune response according to the Danger Theory. A cell that is in distress sends out an alarm signal, whereupon antigens in the neighbourhood are captured by *antigen-presenting cells* such as macrophages, which then travel to the local lymph node and present the antigens to lymphocytes. Essentially, the danger signal establishes a danger zone around itself. Thus B cells producing antibodies that match antigens within the danger zone get stimulated and undergo the clonal expansion process. Those that do not match or are too far away do not get stimulated.

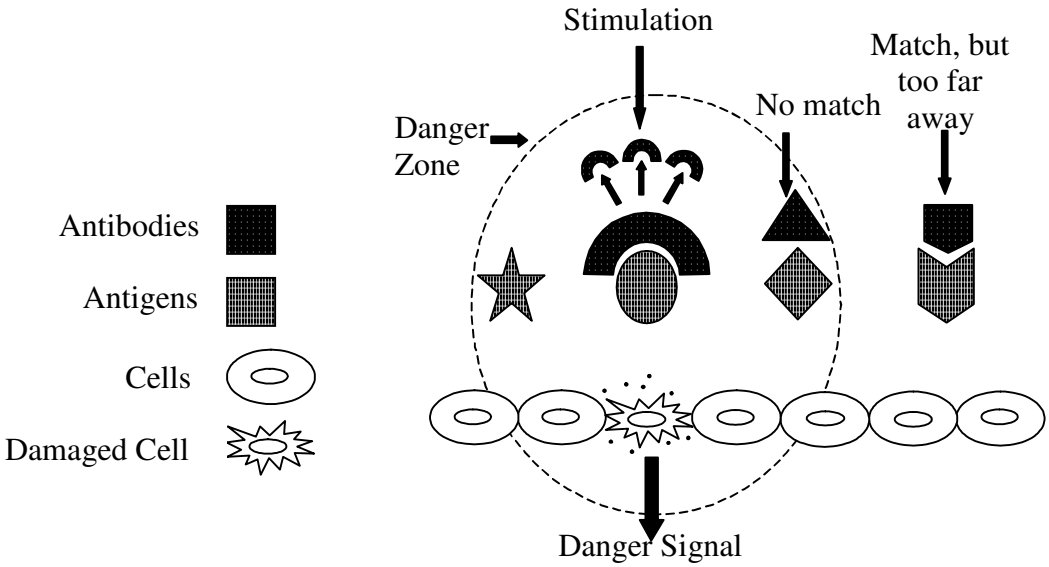


Figure 1: Danger Theory Model.

Matzinger admits that the exact nature of the danger signal is unclear. It may be a 'positive' signal (for example heat shock protein release) or a 'negative' signal (for example lack of synaptic contact with a dendritic antigen-presenting cell). This is where the Danger Theory shares some of the problems associated with traditional self-non-self discrimination (i.e. how to discriminate danger from non-danger). However, in this case, the signal is grounded rather than being some abstract representation of danger.

Another way of looking at the danger model is to see it as an extension of the Two-Signal model by Bretscher and Cohn [4]. In this model, the two signals are antigen recognition (signal one) and co-stimulation (signal two). Co-stimulation is a signal that means "this antigen really is foreign" or, in the Danger Theory, "this antigen really is dangerous". How the signal arises will be explained later. The Danger Theory then operates by applying three laws to lymphocyte behaviour (the *laws of lymphotics* [20]):

- Law 1. Become activated if you receive signals one and two together. Die if you receive signal one in the absence of signal two. Ignore signal two without signal one.
- Law 2. Accept signal two from antigen-presenting cells only (or, for B cells, from T helper cells). B cells can act as antigen-presenting cells only for experienced (memory) T cells. Note that signal one can come from any cells, not just antigen-presenting cells.
- Law 3. After activation (activated cells do not need signal two) revert to resting state after a short time.

For the mature lymphocyte, (whether virgin or experienced) these rules are adhered to. However, there are two exceptions in the lymphocyte lifecycle. Firstly, immature cells are unable to accept signal two from any source. This enables an initial negative selection screening to occur. Secondly, activated (effector) cells respond only to signal one (ignoring signal two), but revert to the resting state shortly afterwards.

An implication of this theory is that autoreactive effects are not necessarily harmful, and are in fact expected during an infection. This is because any lymphocyte reacting to an antigen in the 'danger zone' will be activated. These antigens are not necessarily the culprits for the danger signal. If they are, then the reacting lymphocytes will continue to be restimulated until the antigens (and therefore the danger signal) are removed. After this, they will rest, receiving neither signal one nor signal two.

On the other hand, lymphocytes reacting to innocuous (self) antigens will continue to receive signal one from these antigens, even after the danger (and therefore signal two) has vanished. Therefore these lymphocytes will be deleted, and tolerance will be achieved. However, further autoreactive effects can be expected, partly because 'self' changes over time, and partly because of new lymphocyte generation (particularly B cells, which produce hypermutated clones during activation).

A problem is posed by the antigen-presenting cell itself, whose (innocuous) antigens are by definition always in the danger zone. Lymphocytes reacting to these antigens might destroy the antigen-presenting cell and thus interfere with the immune response. The negative selection of immature lymphocytes protects against this possibility.

Figure 2 shows a more detailed picture of how the Danger Theory can be viewed as an extension of immune signals. These diagrams are adapted from those presented in Matzinger [19] except for the sixth, which incorporates suggestions made in Matzinger [20].

In the original view of the world by Burnet [6], only signal one is considered. This is shown in the first diagram, where the only signal shown is that between infectious agents and lymphocytes (B cells, marked B, and T killer, marked Tk). Signal two (second diagram) was introduced by Bretscher and Cohn [4]. This helper signal comes from a T helper cell (marked Th), on receipt of signal one from the B cell. That is, the B cell presents antigens to the T helper cell and awaits the T cell's confirmation signal. If the T cell recognises the antigen (which, if negative selection has worked, should mean the antigen is non-self) then the immune response can commence. It was Lafferty and Cuningham [17] who proposed that the T helper cells themselves also need to be 'switched on' by signals one and two, both from antigen-presenting cells. This process is depicted in the third diagram.

Note that the T helper cell gets signal one from two sources – the B cell and the antigen-presenting cell. In the former case the antigens are not chosen randomly – the very opposite, since B cells are highly selective for a range of (hopefully non-self) antigens. In the latter case, the antigens are chosen randomly (the antigen-presenting cell simply presents any antigen it picks up) but signal two should only be provided to the T helper cell for non-self antigens. It is not necessarily clear how the antigen-presenting cell 'knows' the antigen is non-self. Janeway [14] introduced the idea of infectious non-self (for example bacteria), which 'primes' antigen presenting cells, i.e. causing signal two to be produced (fourth diagram). This priming signal is labelled as signal 0 in the figures.

Matzinger proposes to allow priming of antigen-presenting cells by a danger signal (fifth diagram). She also proposes to extend the efficacy of T helper cells by routing signal two through antigen presenting cells [20]. We have marked this as 'signal 3' in the sixth diagram (although Matzinger does not use that term, the intention is clear). In Matzinger's words "the antigen seen by the killer need not be the same as the helper; the only requirement is that they must both be presented by the same antigen-presenting cell". This arrangement allows T helper cells to prime many more T killer cells than they would otherwise have been able to.

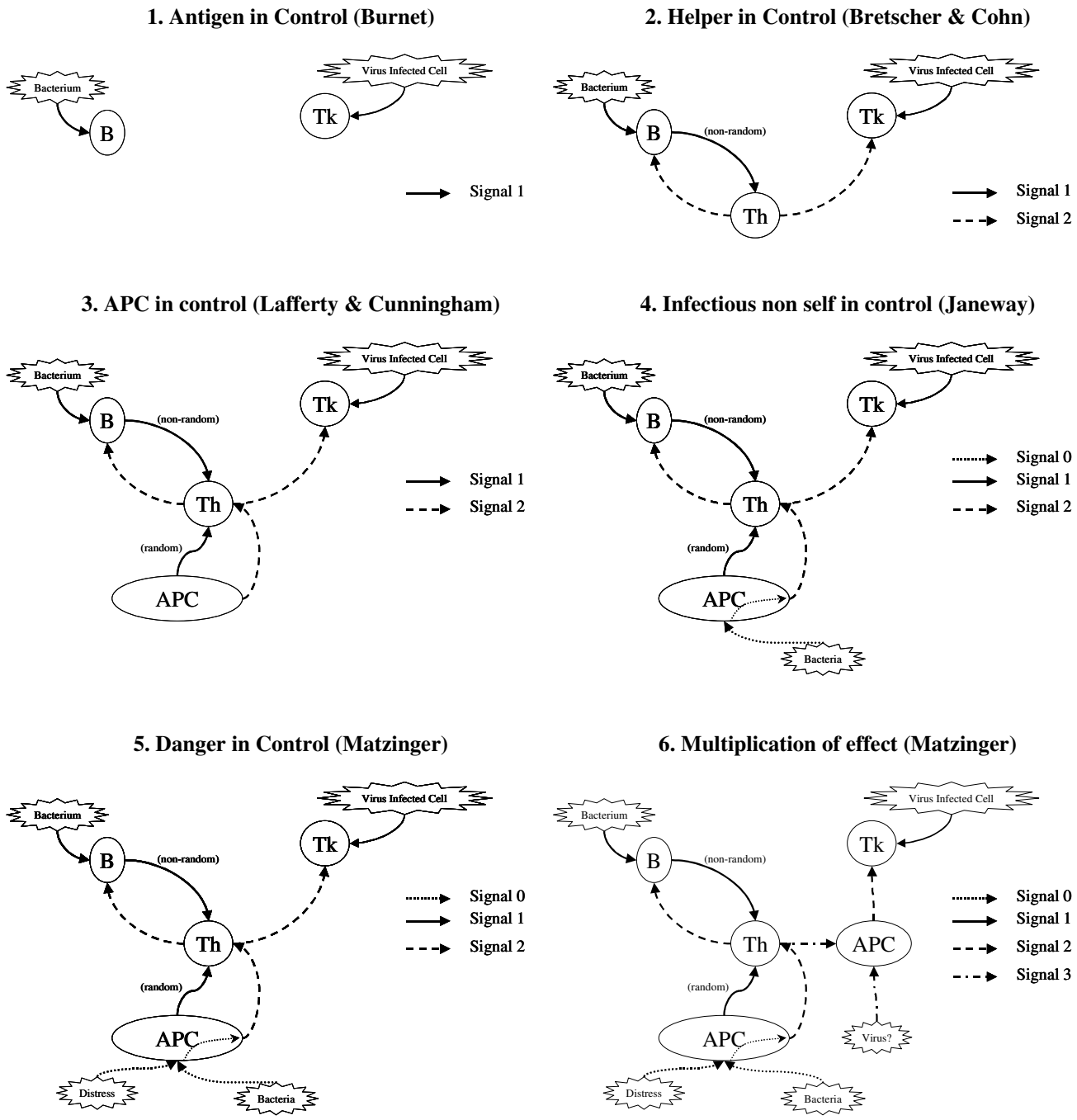


Figure 2: Danger Theory viewed as immune signals.

The Danger Theory is not without its limitations. As mentioned, the exact nature of the danger signal is still unclear. Also, there is sometimes danger that should not be responded to (cuts, transplants). In fact, in the case of transplants it is often necessary to remove the antigen-presenting cells from the transplanted organ. Finally, the fact that autoimmune diseases do still, if rarely, happen, has yet to be fully reconciled with the Danger Theory.

3 THE DANGER THEORY AND SOME ANALOGIES TO ARTIFICIAL IMMUNE SYSTEMS

Danger theory clearly has many facets and intricacies, and we have touched on only a few. It might be instructive to list a number of considerations for an Artificial Immune System practitioner regarding the suitability of the danger model for their application. The basic consideration is

whether negative selection is important. If so, then these points may be relevant:

- Negative selection is bound to be imperfect, and therefore autoreactions (false positives) are inevitable.
- The self/non-self boundary is blurred since self and non-self antigens often share common regions.
- Self changes over time. Therefore, one can expect problems with memory cells, which later turn out to be inaccurate or even autoreactive.

If these points are sufficient to make a practitioner consider incorporating the Danger theory into their model, then the following considerations may be instructive:

1. A danger model requires an antigen-presenting cell, which can present an appropriate danger signal.
2. 'Danger' is an emotive term. The signal may have nothing to do with danger (see, for example, our discussion on data mining applications in section 5).
3. The appropriate danger signal can be positive (presence of signal) or negative (absence).
4. The danger zone in biology is spatial. In Artificial Immune System applications, some other measure of proximity (for instance temporal) may be used.
5. If there is an analogue of an immune response, it should not lead to further danger signals. In biology, killer cells cause a normal cell death, not danger.
6. Matzinger proposes priming killer cells via antigen-presenting cells for greater effect. Depending on the immune system used (it only makes sense for spatially distributed models) this proposal may be relevant.
7. There are a variety of considerations that are less directly related to the danger model. For example, migration – how many antibodies receive signal one/two from a given antigen-presenting cell? In addition, the danger theory relies on concentrations, i.e. continuous not binary matching.

There are also a couple of points that might tempt a practitioner to alter the danger model as presented here. For example, the danger model has quite a number of elements. Given that the antigen-presenting cell mediates the danger signal, we might be able to simplify the model – for example, do we still need a T helper cell? In addition, there are some danger signals that might in some sense be 'appropriate' and thus should not trigger an immune response. In such cases, a method for avoiding the danger pathway must be found. A biological example is transplanted organs, in which antigen-presenting cells are removed.

4 THE DANGER THEORY AND ANOMALY DETECTION

An intriguing area for the application of Artificial Immune Systems is the detection of anomalies such as computer viruses, fraudulent transactions or hardware faults. The underlying metaphor seems to fit particularly nicely here, as there is a system (self) that has to be protected against intruders (non-self). Thus if natural immune systems have enabled biological species to survive, can we not create Artificial Immune Systems to do the same to our computers, machines etc? Presumably those systems would then have the same beneficial properties as natural immune systems like error tolerance, distribution, adaptation and self-monitoring. A recent overview of biologically inspired approaches to this area can be found in Williamson [22].

In this section we will present indicative examples of such artificial systems, explain their current shortcomings and show how the Danger Theory might help overcome some of these.

One of the first such approaches is presented by Forrest et al [11] and extended by Hofmeyr and Forrest [13]. This work is concerned with building an Artificial Immune System that is able to detect non-self in the area of network security where non-self is defined as an undesired connection. All connections are modelled as binary strings and there is a set of known good and bad connections, which is used to train and evaluate the algorithm. To build the Artificial Immune System, random binary strings are created called detectors.

These detectors then undergo a maturation phase where they are presented with good, i.e. self, connections. If they match any of these they are eliminated otherwise they become mature, but not activated. If during their further lifetime these mature detectors match anything else, exceeding a certain threshold value, they become activated. This is then reported to a human operator who decides whether there is a true anomaly. If so the detectors are promoted to memory detectors with an indefinite life span and minimum activation threshold. Thus, this is similar to the secondary response in the natural immune system, for instance after immunisation.

An approach such as the above is known in Artificial Immune Systems as negative selection as only those detectors (antibodies) that do not match live on. It is thought that T cells mature in similar fashion in the thymus such that only those survive and mature that do not match any self cells after a certain amount of time.

An alternative approach to negative selection is that of positive selection as used for instance by Forrest et al [9] and by Somayaji and Forrest [22]. These systems are a reversal of the negative selection algorithm described above with the difference that detectors for self are evolved. From a performance point of view there are advantages and disadvantages for both methods. A suspect non-self string would have to be compared with all self-detectors to establish that it is non-self, whilst with

negative selection the first matching detector would stop the comparison. On the other hand, for a self-string this is reversed giving positive selection the upper hand. Thus, performance depends on the self to non-self ratio, which should generally favour positive selection.

However, there is another difference between the two approaches: the nature of false alarms. With negative selection inadequate detectors will result in false negatives (missed intrusions) whilst with positive selection there will be false positives (false alarms). The preference between the two in this case is likely to be problem specific.

Both approaches have been extended further [10] including better co-stimulation methods and activation thresholds to reduce the number of false alarms, multiple antibody sub-populations for improved diversity and coverage and improved partial matching rules. Recently, similar approaches have also been used to detect hardware faults (Bradley and Tyrrell [1]), network intrusion (Kim and Bentley [16]) and fault tolerance (Burgess [5]).

What are the remaining challenges for a successful use of Artificial Immune Systems for anomaly detection? Firstly, self and non-self will usually evolve and change during the lifetime of the system. Hence, to be effective, any system used must be robust and flexible enough to cope with changing circumstances. Based on the performance of their natural counterparts, Artificial Immune Systems should be well suited to provide these qualities. Secondly, appropriate representations of self and good matching rules have to be developed. Most research so far has been concentrated in these two areas and good advances have been made so far [8].

However, as pointed out by Kim and Bentley [15], scaling is a problem with negative selection. As the systems to be protected grow larger and larger so does self and non-self and it becomes more and more problematic to find a set of detectors that provides adequate coverage whilst being computationally efficient. It is inefficient, if not impossible, to map the entire non-self universe, particularly as it will be changing over time. The same applies to positive selection and trying to map all of self.

Moreover, the approaches so far have another disadvantage: A response requires infection beyond a certain threshold and human intervention confirming this. Although one might argue that the operator sees fewer alarms than in an unaided system, this clearly is not yet the ideal situation of an autonomous system preventing all damage. Apart from the resource implication of a human component, an unduly long delay might be caused by this necessity prolonging the time the system is exposed. This situation might be further aggravated by the fact that the labels self and non-self are often ambiguous and expert knowledge might be required to apply them correctly.

How can these problems be overcome? We believe that applying ideas from the Danger Theory can help building better Artificial Immune Systems by providing a different way of grounding and removing the necessity to map self or non-self. To achieve this self-non-self discrimination will still be useful but it is no longer essential. This is because non-self no longer causes an immune response. Instead, it will be danger signals that trigger a reaction.

What could such danger signals be? They should show up after limited infection to minimise damage and hence have to be quickly and automatically measurable. Suitable signals could include:

- Too low or too high memory usage.
- Inappropriate disk activity.
- Unexpected frequency of file changes as measured for example by checksums or file size.
- SIGABRT signal from abnormally terminated UNIX processes.
- Presence of non-self.

Of course, it would also be possible to use 'positive' signals, as discussed in the previous section, such as the absence of some normal 'health' signals.

Once the danger signal has been transmitted, the immune system can then react to those antigens, for example, executables or connections, which are 'near' the emitter of the danger signal. Note that 'near' does not necessarily mean geographical or physical closeness, something that might make sense for connections and their IP addresses but probably not for computer executables in general. In essence, the physical 'near' that the Danger Theory requires for the immune system is a proxy measure for causality. Hence, we can substitute it with more appropriate causality measures such as similar execution start times, concurrent runtimes or access of the same resources.

Consequently, those antibodies or detectors that match (first signal) those antigens within a radius, defined by a measure such as the above (second signal), will proliferate. Having thereby identified the dangerous components, further confirmation could then be sought by sending it to a special part of the system simulating another attack. This would have the further advantage of not having to send all detectors to confirm danger. In conclusion, using these ideas from the Danger Theory has provided a better grounding of danger labels in comparison to self / non-self, whilst at the same time relying less on human competence.

5 THE DANGER THEORY AND OTHER ARTIFICIAL IMMUNE SYSTEM APPLICATIONS

It is not immediately obvious how the Danger Theory could be of use to data mining problems such as the movie prediction problem described in Cayzer and Aickelin [7], because the notions of self and non-self are not used. In essence, in data mining all of the system is self. More precisely, it is not an issue what is self or non-

self as the designer of the database has complete control over this aspect.

However, if the labels self and non-self were to be replaced by interesting and non-interesting data for example, a distinction would prove beneficial. In this case, the immune system is being applied as a classifier. If one can then further assume that interesting data is located 'close' or 'near' to other interesting data, ideas from the Danger Theory can come into play again. To do so, it is necessary to define 'close' / 'near'. We could use:

- Physical closeness, for instance distance in the database as measured by an appropriate metric.
- Correlation of data, as measured by statistical tools.
- Similar entry times into the database.
- File size.

A danger signal could thus be interpreted as a valuable piece of information that has been uncovered. Hence, those antibodies are stimulated that match data that is 'close' this valuable piece of information.

Taking this idea further, we might define the danger signal as an indication of user interest. Given this definition, we can speculate about various scenarios in which the danger signal could be of use. One such scenario is outlined below for illustrative purposes.

Imagine a user browsing a set of documents. Each document has a set of features (for instance keywords, title, author, date etc). Imagine further that there is an immune system implemented as a 'watcher', whose antibodies match document features. 'Interesting' documents are those, whose features are matched by the immune system.

When a user either explicitly or implicitly indicates interest in the current document, a "danger" signal is raised. This causes signal two to be passed, along with signal one, to antibodies matching any antigen, i.e. document feature, in the danger zone, i.e. this document.

Stimulated antibodies become effectors, and thus the immune system learns to become a good filter when searching for other interesting documents. Interesting documents could be brought to the user's attention (the exact mechanism is not relevant here). The important thing is that the user's idea of an 'interesting' document may change over time and so it is important that the immune system adapts in a timely way to such a changing definition of (non-) self.

Meanwhile, every document browsed by the user (whether interesting or not) will be presented to the antibodies as 'signal one'. Uninteresting document features will therefore give rise to signal one without signal two, which will tolerate the autoreactive antibodies. The net effect is to produce a set of antibodies that match only interesting document features.

As mentioned, this example is purely illustrative but it does show that ideas from the Danger theory may have implications for Artificial Immune System applications in domains where the relevance of 'danger' is far from obvious.

6 CONCLUSIONS

To conclude, the Danger Theory is not about the way Artificial Immune Systems represent data. Instead, it provides ideas about which data the Artificial Immune Systems should represent and deal with. They should focus on dangerous, i.e. interesting data.

It could be argued that the shift from non-self to danger is merely a symbolic label change that achieves nothing. We do not believe this to be the case, since danger is a grounded signal, and non-self is (typically) a set of feature vectors with no further information about their meaning. The danger signal helps us to identify which subset of feature vectors is of interest. A suitably defined danger signal thus overcomes many of the limitations of self-non-self selection. It restricts the domain of non-self to a manageable size, removes the need to screen against all self, and deals adaptively with scenarios where self (or non-self) changes over time.

The challenge is clearly to define a suitable danger signal, a choice that might prove as critical as the choice of fitness function for an evolutionary algorithm. In addition, the physical distance in the biological system should be translated into a suitable proxy measure for similarity or causality in an Artificial Immune System. We have made some suggestions in this paper about how to tackle these challenges in a variety of domains, but the process is not likely to be trivial. Nevertheless, if these challenges are met, then future Artificial Immune System applications might derive considerable benefit, and new insights, from the Danger Theory.

Acknowledgements

We would like to thank the two anonymous reviewers, whose comments greatly improved this paper.

References

[1] Bradley D, Tyrell A, The Architecture for a Hardware Immune System, *Proceedings of the 2002 Congress on Evolutionary Computation*, 2002.

[2] Forrest H. Bennett III, John R. Koza, Jessen Yu, William Mydlowec, Automatic Synthesis, Placement, and Routing of an Amplifier Circuit by Means of Genetic Programming. *Evolvable Systems: From Biology to Hardware, Third International Conference, ICES 2000:* 1-10, 2000

[3] D. W. Bradley, Andrew M. Tyrrell, Immunotronics: Hardware Fault Tolerance Inspired by the Immune System. *Evolvable Systems: From Biology to Hardware, Third International Conference, ICES 2000:* 11-20, 2000

[4] Bretscher P, Cohn M, A theory of self-nonself discrimination, *Science* 169, 1042-1049, 1970

[5] Burgess M: Computer Immunology, *Proceedings of LISA XII*, 283-297, 1998.

[6] Burnet F, *The Clonal Selection Theory of Acquired Immunity*, Vanderbilt University Press, Nashville, TN, 1959.

[7] Cayzer S, Aickelin U, A Recommender System based on the Immune Network, *Proceedings of the 2002 Congress on Evolutionary Computation*, 2002.

[8] Dasgupta D, Majumdar N, Nino F, Artificial Immune Systems: A Bibliography, *Computer Science Division, University of Memphis, Technical Report* No. CS-02-001, 2001.

[9] Forrest S, Hofmeyr S, Somayaji A, Longstaff T, A sense of self for Unix processes, *Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy*, 120-128, 1996.

[10] Forrest S, http://www.cs.unm.edu/~immsec/, 2002.

[11] Forrest S, Perelson A, Allen L, Cherukuri R, Self-non-self discrimination in a computer, *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, 202-212, 1994.

[12] Goldsby R, Kindt T, Osborne B, *Kuby Immunology*, Fourth Edition, W H Freeman, 2000.

[13] Hofmeyr S, Forrest S, Architecture for an Artificial Immune System, *Evolutionary Computation* 8(4), 443-473, 2000.

[14] Janeway C, The immune System evolved to discriminated infectious nonself from noninfectious self, *Immunology Today* 13, 11-16, 1992.

[15] Kim J, Bentley P, An evaluation of negative selection in an artificial immune for network intrusion detection, *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, 1330-1337, 2001.

[16] Kim J, Bentley P, Towards an Artificial Immune System for Network Intrusion Detection: An Investigation of Dynamic Clonal Selection. *Proceedings of the 2002 Congress on Evolutionary Computation*, 2002.

[17] Lafferty K, Cunningham A, A new analysis of allogeneic interactions. *Australian Journal of Experimental Biology and Medical Sciences*. 53:27-42, 1975

[18] Matzinger P, http://cmmg.biosci.wayne.edu/asg/polly.html

[19] Matzinger P, The Danger Model in Its Historical Context, *Scandinavian Journal of Immunology*, 54: 4-9, 2001.

[20] Matzinger P, Tolerance, Danger and the Extended Family, *Annual Review of Immunology*, 12:991-1045, 1994.

[21] Langman R (editor), Self non-self discrimination revisited, *Seminars in Immunology* 12, Issue 3, 2000.

[22] Somayaji A, Forrest S, Automated response using system-call delays, *Proceedings of the ninth USENIX Security Symposium*, 185-197, 2000.

[23] Williamson M, Biologically inspired approaches to computer security, HP Labs Technical Reports HPL-2002-131, 2000 (available from http://www.hpl.hp.com/techreports/2002/HPL-2002-131.html).

Artificial Immune Systems for Classification : Some Issues

Gaurav Marwah
Department of Computer Science
Mississippi State University

Lois Boggess
Department of Computer Science
Mississippi State University

Abstract

It has recently been shown that Artificial Immune Systems are not only capable of performing classification, but that AIRS, a resource limited Artificial Immune System, is competitive with some of the best classifiers in the world on a broad variety of classification problems. This paper explores some of the issues that affect the performance of AIRS. These include modifications to the algorithm for resource allocation, a policy for handling ties and approaches to ARB pool organization.

1 Introduction

AIRS (Artificial Immune Recognition System) (Watkins, 2001) is a classifier developed from the principles of resource limited artificial immune systems (Timmis and Neal, 2001), which have already been shown to be effective clustering tools.

AIRS has been applied to a wide variety of publicly available classification benchmarks (Watkins, 2001), (Watkins and Boggess, 2002a), (Watkins and Boggess, 2002b), (Goodman, Boggess and Watkins, 2002). Although the initial objective was simply to show that artificial immune systems could be used as classifiers, AIRS proved to be a very good classifier indeed: thus far it has been among the ten most accurate classifiers known in every case to which it has been applied, with only one exception. Often it outperforms some of the best known classifiers in general use. In one classification problem from the UCI machine learning repository (Blake and Merz, 1998), AIRS's average performance appears to edge out the best reported classification results (Goodman, Boggess and Watkins , 2002).

This performance is intriguing for several reasons: AIRS is self-regulatory in that it is not necessary for a user to know in advance the best architecture or best set of parameters for AIRS to perform well. Modifying the user-adjustable parameters of the system allows AIRS to be fine-tuned to a given problem domain. Nevertheless with no fine-tuning at all, AIRS still tends to perform within a few percentage points of its optimum for the given domain. Moreover, this is a new classifier, so it seems reasonable that further study and development may lead to improvements and even better results.

This paper explores some aspects of AIRS that might affect its performance as a classifier. The remainder of the paper is structured as follows; section 2 gives an overview of AIRS. Section 3 discusses some of the experiments performed on AIRS. This is followed by conclusions, acknowledgments and references in that order.

2 AIRS

AIRS is modeled mainly on the mechanisms followed by the B-cells of the biological immune system. Antigens in AIRS are instantiated as feature vectors which are presented to the system during training and testing. B-cells in AIRS follow the same representation as antigens. All the B-cells having similar features are represented together as ARBs (Artificial Recognition Balls).

AIRS is a resource-bounded supervised learning system. ARBs compete for a fixed number of resources; this helps in gradual evolution of those ARBs which represent training antigens more closely. Another component of AIRS is the pool of memory cells, which are similar to B-cells except that they have an extended life span and are used for actual classification of test antigens. A pool of ARBs is used for breeding candidate memory cells. The mechanism to develop a candidate memory cell is as follows:

1. A training antigen is presented to all the memory cells belonging to the same class as the antigen. The memory cell most stimulated by the antigen is cloned. The memory cell and all the just-generated clones are put into the ARB pool. The number of clones generated depends on the affinity between the memory cell and antigen, and affinity in turn is determined by Euclidean distance between the feature vectors of the memory cell and the training antigen. The smaller the Euclidean distance, the higher the affinity, the more is the number of clones allowed.
2. Next, the training antigen is presented to all the ARB's in the ARB pool. All the ARB's are appropriately rewarded based on affinity between the ARB and the antigen as follows: An ARB of the same class as the antigen is rewarded highly for high affinity with the antigen. On the other hand, an out of class ARB is rewarded highly for a low value of affinity measure. The rewards are

in the form of number of resources. After all the ARBs have been rewarded, the sum of all the resources in the system typically exceeds the maximum number allowed for the system. The excess number of resources held by ARB's are removed in order starting from the ARB of lowest affinity and moving higher until the number of resources held does not exceed the number of resources allowed for the system. Those ARBs, which are not left with any resources, are removed from the ARB pool. The remaining ARBs are tested for their affinities towards the training antigen. If for any class of ARB the total affinity over all instances of that class does not meet a user defined stimulation threshold, then the ARBs of that class are mutated and their clones are placed back in the ARB pool. Step 2 is repeated until the affinity for all classes meet the stimulation threshold.

3. After ARBs of all classes have met the stimulation threshold, the best ARB of the same class as the antigen is chosen as a candidate memory cell. If its affinity for the training antigen is greater than that of the original memory cell selected for cloning at step 1, then the candidate memory cell is placed in the memory cell pool. If in addition to this the difference in affinity of these two memory cells is smaller than a user defined threshold, the original memory cell is removed from the pool.

These steps are repeated for each training antigen. After completion of training the test data are presented only to the memory cell pool, which is responsible for actual classification. The class of a test antigen is determined by majority voting among the k most stimulated memory cells, where k is a user defined parameter.

3 Experiments

3.1 Handling Ties

The current version of AIRS follows a k nearest neighbor voting scheme for classification. This means that majority voting among the k most stimulated memory cells determines the class of the test antigen. For a k value of 1 only the most stimulated memory cell is used for prediction. An important question that comes to mind for higher values of k is "What should be done in case of a tie?" That is, how should the class be determined when two or more classes have an equal number of memory cells among the k strongest stimulated memory cells? One straightforward answer to this might be that nothing needs to be done: since k is a training parameter, a different value of k may be chosen which might result in fewer ties.

Though this approach may prove successful in many cases, there is no guarantee that a value of k which minimizes ties will also maximize accuracy. This is because the parameter k affects both the accuracy and the number of ties independently.

Since there is no inherent reason to suppose that changing the value of k is likely to be the best way to handle ties, we therefore propose a number of alternatives for handling ties. The effect of these approaches on the accuracy of the classifier varies depending on the characteristics of the problem.

The need for these alternatives was realized while testing AIRS on the well-known and publicly available yeast data set, which appears to be a difficult classification problem. The data set was obtained from the repository of the University of California at Irvine (Blake and Merz, 1998) and contained 1484 instances representing ten classes. Ten-way cross validation was performed and the best accuracy rate obtained after optimal setting of parameters was 51.23%. For purposes of comparison, the original donor of that dataset to UCI, Paul Horton, after extensive experimentation on that classification problem reported a best accuracy of about 60%, with a best previously known accuracy of 55% (Horton and Nakai, 1997).

Table 1 and Table 2 show some of the characteristics of the problem, which suggest that it is the distinction between the first three classes that poses the maximum challenge to classifiers.

Table 1: Confusion Matrix for a Test Run on Yeast Data

20	11	6	0	0	0	0	0	0	0
22	15	5	0	0	0	0	0	0	0
8	0	7	0	0	0	0	0	0	0
1	3	0	4	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	7	0	0	0	0
1	0	0	0	0	2	0	0	0	0
1	1	1	1	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0	0	0

Table 2. Collision Matrix for Classification Ties in Test Run on Yeast Data

0	7	1	1	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

The confusion matrix and collision matrix shown in Tables 1 and 2 were obtained using a single training (1364 instances) and test set (120 instances) of yeast data set on AIRS. The contents of the matrices shown in the above tables are for the test data only. The value in row i, column j of a confusion matrix represents the number of instances of class i that were classified as belonging to class j. All the diagonal entries therefore represent test instances that were correctly classified. All other entries represent instances of incorrect classification. As an example, the first entry in row 2 of Table 1 shows that there were 22 test instances belonging to class 2 that were wrongly classified as belonging to class 1. Similarly, the first entry in row 3 shows that 8 test instances of class 3 were wrongly classified as class 1. Indeed, it is the first three classes which cause maximum inaccuracy. In part, this is because the instances of these classes occur in maximum proportions.

The nature of the yeast classification problem is such that the given features are not sufficient to distinguish between classes. However, inaccuracy due to wrong prediction of class in case of a tie between two or more classes can be better handled. Table 2 shows the content of what we call the collision matrix that helps to understand this situation. An entry in row i, column j of the collision matrix represents the number of times an actual member of class j was involved in a tie with class i and lost it. It shows that class 2 lost to class 1 on seven occasions whereas class 1 never lost to class 2. The matrices shown are for only one instance of training and test sets; nevertheless it is representative of our observation over many runs that only the upper triangular region of collision matrix contained nonzero entries. This was found to be because of the approach followed by the original AIRS algorithm for tie breaking, which is described next.

AIRS handles ties on a first labeled first served basis. This means that in case of a tie, the class that was labeled earlier wins the tie. For example, in the case of the yeast problem, class 1 will never lose the tie and class 10 will always lose the tie. This is the reason behind high values in the first row of the collision matrix. This tie-breaking scheme may be appropriate for a k nearest neighbor-style classifier if the basic classification algorithm is modified to ensure that classes are labeled in decreasing order of relative proportion, for in that case a tie will always go in favor of the class with higher proportional representation. One drawback however is that such a tie-breaking rule never rules in favor of the less frequent class.

We suggest multiple methods for handling ties. Because different methods may work best for different classification problems, we suggest that the actual choice of method be used as a training parameter.

Four different approaches for tie breaking were tried and are described next.

1. Sum of affinities:

This method uses the sum of affinities of memory cells of the same class among the k strongest stimulated memory cells. The class with highest combined affinity represents the predicted class for the test antigen. One feature of this method is that it completely replaces the typical k nearest neighbor voting scheme, and is always used irrespective of an actual tie being present or not. Because it involves floating point values the likelihood of a tie is extremely remote using this approach.

Use of sum of affinities gives more influence to highly stimulated memory cells, and the relative effect of less stimulated memory cells is low.

2. Selection based on class proportions:

Unlike the sum of affinities method, this method is utilized only in case of an actual tie. It predicts the class of the test antigen based on class proportions of competing classes in the training set. For example, in a two way tie if one of the classes occurred twice as often as the other during the training process, than the chances of the tie going in favor of that class will be twice that of the other. This approach is expected to perform better than other approaches for noisy classification problems, including the yeast classification problem.

3. Including more memory cells.

This method involves looking at more memory cells in case of a tie. Memory cells beyond the k strongest stimulated ones are also incorporated into the vote one at a time until the tie is broken or the number of additional memory cells exceeds some predefined value, *k_additional*, which can be set as a fixed proportion of k. In case the tie is not broken and the number of additional cells becomes equal to *k_additional*, the selection approach described in method 2 may be used.

This approach effectively increases the k value for those test cases where there is a tie while leaving it the same for other instances. Consequently, there is some separation between the value of k and its effect on accuracy and on tie breaking. This approach may be modified so that instead of using additional memory cells, fewer memory cells may be used until the tie is broken.

4. First come first served.

This method uses the affinities of the most stimulated memory cell of each of the competing classes to decide the tie. What this means is that once a tie has occurred among two or more classes, the class which has the highest stimulated memory cell is chosen.

Table 3 shows accuracy rates obtained for the yeast data set using different alternatives for handling ties. These results were obtained using the same set of AIRS parameters and a single test and training set. Class prediction based on relative class proportions seems to work best for this case as expected. Also, prediction based on looking for more memory cells in case of a tie works better than the original approach followed by AIRS.

The reason behind the better performance of "selection based on class proportion" is that it probabilistically breaks the tie based on relative frequency of occurrence of various classes during training cycle. As such, it is

expected to work well for noisy classification problems in which the relative frequency of classes may be the only useable data in the toughest classification regions of the problem space.

Table 3: Accuracy Rates For Yeast Data Set Using Different Approaches For Tie Breaking

Method	Accuracy
First labeled first served	46.67 %
Sum of affinities	44.17 %
Selection based on class proportions	48.33 %
Including more memory cells	47.5 %
First come first served	44.72 %

The "sum of affinities" method does not do particularly well for this case. This method will do well if the memory cells representing the true classes are more highly stimulated in the aggregate than their equally numerous but less highly stimulated competitors in the tie-breaking region.

The "First come first served" approach is highly biased in favor of highly stimulated memory cells. This approach will be helpful when the best memory cells of competing classes in the tie-breaking region are representative of their classes for that region.

3.2 ARB Pool Reorganization

As already described, AIRS uses the memory cell pool for actual classification of test antigens, whereas the ARB pool is used along with the memory cell pool during training. The ARB pool is used as a breeding ground for candidate memory cells.

Watkins (personal communication) modified the ARB pool organization so that the most recent version of AIRS no longer keeps track of ARBs from previous training instances. The reason behind this can be explained as follows:

During training on an antigen, the stimulation threshold is used as a stopping criterion. Training on an antigen is continued until all the classes satisfy the requirements of the stimulation threshold: for all ARBs of the same class as the antigen their affinity for the current training antigen should be greater than the user-defined stimulation threshold. In addition, for all out of class ARBs the affinity should be less than (1 – stimulation threshold). For example, for a stimulation threshold of 0.8, all in-class ARBs should have affinities with the training antigen greater than 0.8, and all out of class ARBs should have affinities less than 0.2. The eventual effect of this process is that at the end of training the out of class ARBs remaining are extremely distant from the training instance and not very useful for producing candidate memory cells during training on subsequent antigens.

Therefore, in the modified version of AIRS (Watkins, personal communication), all ARBs from previous training are removed, and only the most simulated memory cell along with its clones is allowed in the ARB pool. This modification does not seem to affect the accuracy of the classifier for some of the problems on which it has been tested (Watkins, personal communication). However, it greatly decreases the diversity in the ARB pool. We tried two other approaches for ARB pool reorganization.

1. In the first approach, the ARBs from previous training stages were allowed to exist and compete for resources in the ARB pool; however affinities between out of class ARBs and the current antigen were not considered. What this means is that ARBs from previous instances did not undergo mutation and therefore represented previously seen antigens and any memory cells developed for them more closely.

2. In the second approach, the ARBs from previous stages were allowed to exist and compete for resources. In addition, ARBs of all classes were required to have affinities with the training antigen satisfying some stimulation threshold, but the stimulation threshold for out of class ARBs was somewhat relaxed as compared to in class ARBs.

Table 4 shows the accuracy rates obtained for the iris data set using the approaches just described. Five way cross validation was performed to achieve these results.

Table 4: Accuracy Rates For Iris Data Set Using Different Approaches For ARB Pool Organization

Scheme for ARB pool organization	Accuracy
Competition between ARBs of different classes and affinities for all classes of ARB to satisfy stimulation threshold.	96.7 %
Competition for resources only, affinities between out of class ARBs and antigen not considered.	95.56 %
Competition for resources with relaxed condition regarding affinities for out of class ARBs.	96.23 %

At least for the present, our results support the assumption that diversity of out of class ARBs in the ARB pool is not a significant factor in quality of memory cells generated.

3.3 Resource Allocation

As already mentioned, AIRS is a resource bounded supervised learning system. A fixed number of resources are distributed among ARBs based on their affinities for current training antigen. This competition of resources helps in evolution of ARBs that represent training antigens more and more closely. The scheme used by AIRS for resource allocation is that half of the resources are distributed among ARBs of same class as the current training antigen, and the other half is distributed among ARBs of other classes.

Another approach that was tried distributed resources based on class proportions obtained from training data. The classes of antigen occurring more frequently were allocated more resources and those occurring less frequently were allocated fewer resources.

Table 5: Accuracy Rates For E.coli And Yeast Data Sets Using Different Methods For Resource Allocation.

Method used for resource allocation	Accuracy (E.Coli)	Accuracy (Yeast)
Half the resources for in class ARBs and the other half for out of class ARBs.	85.71 %	52.23 %
Resource allocation based on class proportions.	86.30 %	51.08 %

Table 5 shows the accuracy rates obtained using the two approaches for resource allocation for the yeast and E.coli data sets. Five way cross validation was performed for the E.coli data set and for the yeast data set ten fold cross validation was performed. The results are averaged over three runs for each case. It shows a marginal increase in accuracy for the E.coli data set using resource allocation based on class proportion; however for the yeast data set accuracy decreased using this approach. We point out in passing that the best reported accuracy at the UCI web site for the E.coli data set is 81% (Blake and Merz, 1998).

4 Conclusions

Since AIRS is a very recent classifier, there are many possible areas for exploration of the algorithm. In this paper we have explored several different algorithms for tie breaking which could increase the accuracy of AIRS and other k nearest neighbor classifiers, especially for tougher classification problems. On the other hand, variations on resource allocation and ARB pool organization which were investigated were mixed or ineffective in improving the original AIRS algorithm. In the course of these investigations, AIRS produced a higher average accuracy for one of the testbeds than any reported at the UCI repository for that testbed.

Acknowledgments

We are grateful to Andrew Watkins for providing his code for AIRS and some of the data sets. We would also like to extend thanks to Don Goodman for providing his data set and scripts, which were very useful in better understanding of the problem.

References

Blake, C.L and C.J Merz. 1998. UCI Repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html. University of California at Irvine, Department of Computer Science.

Goodman, D., L. Boggess and A. Watkins, 2002. Artificial Immune System Classification of Multiple-Class Problems. Artificial Neural Networks In Engineering, ANNIE 2002 (accepted).

Horton, P. and Kenta Nakai. 1997. Better Prediction of Protein Cellular Localization Sites with k Nearest Neighbors Classifer. In Proceedings, Intelligent Systems in Molecular Biology, 368-383.

Timmis, J. and M Neal. 2001. A Resource Limited Artificial Immune System for Data Analysis. Knowledge Based Systems 14(3-4): 121-130.

Watkins, A. 2001. AIRS: A Resource Limited Artificial Immune Classifier. M.S Thesis, Department of Computer Science, Mississippi State University.

Watkins, A. and L. Boggess. 2002a. A New Classifier Based on Resource Limited Artificial Immune Systems. In Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002), IEEE Press. Vol 2: 1546-1551.

Watkins, A. and L. Boggess. 2002b. A Resource Limited Artificial Immune Classifier. In Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002), Special Session on Artificial Immune Systems, IEEE Press. Vol 1: 926-931.

On the Effects of Idiotypic Interactions for Recommendation Communities in Artificial Immune Systems

Steve Cayzer
Hewlett-Packard Laboratories
Filton Road
Bristol
BS12 6QZ
Steve_Cayzer@hp.com

Uwe Aickelin
Department of Computing
University of Bradford
Bradford
BD7 1DP
u.aickelin@bradford.ac.uk

Abstract

It has previously been shown that a recommender based on immune system idiotypic principles can outperform one based on correlation alone. This paper reports the results of work in progress, where we undertake some investigations into the nature of this beneficial effect. The initial findings are that the immune system recommender tends to produce different neighbourhoods, and that the superior performance of this recommender is due partly to the different neighbourhoods, and partly to the way that the idiotypic effect is used to weight each neighbour's recommendations.

1 INTRODUCTION

The idiotypic effect builds on the premise that antibodies can match other antibodies as well as antigens. It was first proposed by Jerne [6] and formalised into a model by Farmer et al [3]. The theory is currently debated by immunologists, with no clear consensus yet on its effects in the humoral immune system [5]. In a previous paper [1], we have shown that the incorporation of idiotypic effects can be beneficial for Artificial Immune System based recommender systems.

However, in that paper we did not explore the mechanisms of that beneficial effect. Such an exploration would seem worthwhile, particularly if this results in identifying the underlying causes of the improvements of the 'characteristics' of a community (either by changing its membership, or by evaluating the relative merit of each member). Such an effect will be generally useful in a range of applications, of which recommender systems provide just one example. In addition, a deeper understanding of the idiotypic effect may prove useful to the designers of other Artificial Immune System applications.

In this paper, we present the results of work undertaken to better understand the idiotypic effect. In order to set the context, the next section provides a definition of the idiotypic effect and the following one a brief review of Artificial Immune System based recommenders. We then present and discuss the results of our analysis to date.

2 IDIOTYPIC EFFECTS

The idiotypic network hypothesis was first proposed by Jerne [6]. It builds on the recognition that antibodies can match other antibodies as well as antigens. Hence, an antibody may be matched by other antibodies, which in turn may be matched by yet other antibodies. This activation can continue to spread through the population. The idiotypic network has been formalised by a number of theoretical immunologists in [7]. This theory could help explain how the memory of past infections is maintained. Furthermore, it could result in the suppression of similar antibodies thus encouraging diversity in the antibody pool.

The following is a formal equation for the idiotypic effect adapted from Equation 3 from Farmer [3]:

$$\frac{dx_i}{dt} = c\left[\begin{pmatrix}antibodies\\recognised\end{pmatrix} - \begin{pmatrix}I\,am\\recognised\end{pmatrix} + \begin{pmatrix}antigens\\recognised\end{pmatrix}\right] - \begin{pmatrix}death\\rate\end{pmatrix}$$

$$= c\left[\sum_{j=1}^{N} m_{ji}x_i x_j - k_1\sum_{j-1}^{N} m_{ij}x_i x_j + \sum_{j=1}^{n} m_{ji}x_i y_j\right] - k_2 x_i \qquad (1)$$

Where:

N is the number of antibodies

n is the number of antigens.

x_i (or x_i) is the concentration of antibody i (or j)

y_i is the concentration of antigen j

c is a rate constant

k_1 is a suppressive effect and k_2 is the death rate

m_{ji} is the matching function between antibody i and antibody (or antigen) j

As can be seen from the above equation, the nature of an idiotypic interaction can be either positive or negative. Moreover, if the matching function is symmetric, then the balance between "I am recognised" and "Antibodies recognised" (parameters c and k_1 in the equation) wholly determines whether the idiotypic effect is positive or negative, and we can simplify the equation. We can simplify the equation still further if we only allow one antigen in the Artificial Immune System. The simplified equation looks like this:

$$\frac{dx_i}{dt} = k_1 m_i x_i y - \frac{k_2}{n} \sum_{j=1}^{n} m_{ij} x_i x_j - k_3 x_i \qquad (2)$$

Where:

k_1 is stimulation, k_2 suppression and k_3 death rate

m_i is the correlation between antibody i and the (sole) antigen

x_i (or x_i) is the concentration of antibody i (or j)

y is the concentration of the (sole) antigen

m_{ij} is the correlation between antibodies i and j

n is the number of antibodies.

3 RECOMMENDER SYSTEM

At this point, it is worth reviewing how this model can be applied to recommender systems. Full details can be found in [1], but a brief overview follows.

Recommender systems are those that use collaborative filtering techniques to produce predictions and recommendations [4]. So for example a movie recommender system would, given a film, provide a *prediction* for that film (i.e. an estimated rating for you). It might also provide a list of *recommended* films (i.e. films which it estimates that you would prefer over others). It does this by comparing users together (based on their votes for movies), and preparing some 'neighbourhood' of like-minded users from which it can produce predictions and recommendations.

The main loop of the recommender algorithm is shown in Figure 1 and is the core of our Artificial Immune System. The aim of this algorithm is to increase the concentrations of those antibodies (database users) that are similar to the antigen (target user) and yet different from each other. The process is thus subject to the suppression of similar antibodies following Jerne's idiotypic ideas mentioned above. Thus, over time the Artificial Immune System contains high concentrations of a diverse set of users who have similar film preferences to the target user.

The algorithm is terminated either when there are no more users to try, or when the Artificial Immune System is *stabilised*, i.e. it is full, and has not changed in consistency for more than ten iterations. The concentrations and correlations of the users in the final neighbourhood, i.e. final immune system iteration, are then used to calculate a weighted sum of the ratings of movies.

```
Initialise Artificial Immune System
Encode user for whom to make predictions as antigen Ag
WHILE (Artificial Immune System not stabilised) & (More data available) DO
  Add next user as an antibody Ab
  Calculate matching score between Ab and Ag
  Calculate matching scores between Ab and other antibodies
WHILE (Artificial Immune System at full size) & (Artificial Immune System not stable) DO
        Iterate Artificial Immune System
  OD
OD
```

Figure 1: Main loop of the Artificial Immune System's algorithm for recommendation.

Our previous work [1] compared two predictors, one based on a Simple Pearson test and one on our Artificial Immune System. In each case, a test user is taken from a database, and then predictions and recommendations are made for that user. Both predictors work by finding a neighbourhood and using that neighbourhood to produce predictions and recommendations.

Prediction quality is assessed by measuring the mean absolute error (details in [1]). Recommendation quality is assessed by comparing the ranked recommendations with the user's ranked ratings for the recommended films. Kendall's Tau can now be applied. This measure reflects the level of concordance in the lists, and proceeds by counting the number of discordant pairs. To do this we order the films by actual vote and apply the following formulae to the recommended films:

$$\tau = 1 - \frac{4N_D}{n(n-1)}$$

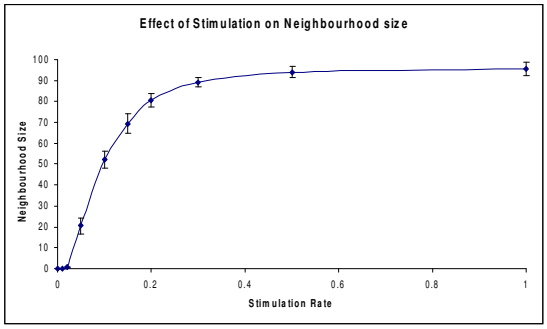
$$N_D = \sum_{i=1}^{n} \sum_{j=i+1}^{n} D(r_i, r_j) \qquad (3)$$

$$D(r_i, r_j) = \begin{cases} 1 if\ r_i > r_j \\ 0 otherwise \end{cases}$$

Where:
n is the overlap size
r_i is the actual rank of film i as recommended by the neighbourhood.

Note that i here refers to the recommended rank of the film, not the film ID. N_D is the number of discordant pairs, or, equivalently, the expected cost of a bubble sort to reconcile the two lists. D is set to one if the rankings are discordant.

2a)



(2b)

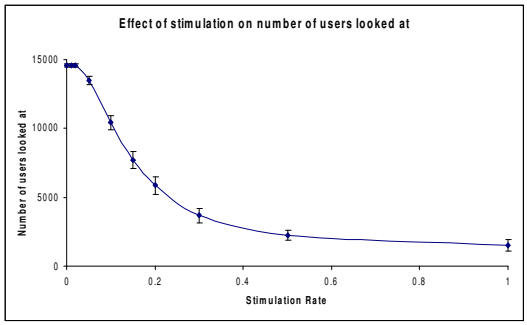


Figure 2: Effect of stimulation rate on neighbourhood size and reviewers looked at.

For the Simple Pearson case, the neighbourhood is composed of the 'top N' correlated users, where correlation is measured by the Simple Pearson statistical measure. In the Artificial Immune System case, the neighbourhood is created by building an immune system with the test user as the antigen, the neighbours as antibodies, and the Simple Pearson measure as a matching function. (In fact, in our experiments, this measure was weighted by the a fraction proportional to the number of films both users had seen, in order to penalise correlations made on the basis of only a few films). The behaviour of the neighbourhood is then governed by equation 2, with poorly performing antibodies being deleted from the neighbourhood. Note that we have treated the idiotypic effect as suppressive.

4 ANALYSIS OF EFFECTS

Although both the Artificial Immune System and Simple Pearson recommender algorithms are based on Pearson correlations, they act differently for a number of reasons:

- The choice of neighbours is different. In the Simple Pearson, the 100 highest correlated users (or all users that show any correlation, if this is less than 100) are chosen to form a neighbourhood. In the Artificial Immune System, this general rule is followed, except that stimulation adds threshold and idiotypic effect adds diversity.
- Even given the same neighbours, the weighting is different. In the Simple Pearson, the neighbour weight is simply the correlation between that neighbour and the test user. In the Artificial Immune System, this correlation is multiplied by that antibody's (neighbour's) concentration, which in turn is determined by running the Artificial Immune System algorithm over the neighbourhood.

To deal with the first point, the stimulation rate provides some fixed threshold for the correlation of any antibody with the antigen. Even in the absence of any idiotypic interactions, an antibody's correlation (weighted by the stimulation rate) must outweigh the death rate; otherwise, it will not survive in the Artificial Immune System. So, at low stimulation rates it may prove difficult to fill the Artificial Immune System completely. Conversely, at very high stimulation rates it may not be necessary to examine all the supplied users in order to fill an Artificial Immune System.

This effect was noted in our previous paper [1] and can be seen in Figure 2. Such a thresholding effect has been shown to be beneficial by Gokhale [4] in maintaining the quality of a neighbourhood by filtering out poorly correlated users (the Simple Pearson will consider all reviewers who have at least one vote in common with the test user).

Thus, the idiotypic effect should be viewed in the context of providing further refinement to a neighbourhood that is already known to be in some sense 'good'. Since the effect (in our model) is always negative, its impact may be to improve diversity by removing 'suboptimal' users from the Artificial Immune System. Conversely, it might be that the idiotypic effect is effective because, given a neighbourhood, it changes the weight of each neighbour (or concentration of each antibody) in that neighbourhood. This is the second point highlighted above.

In order to test out these hypotheses, we took a sample result, based on 100 predictions for detailed analysis. The 3 settings for each algorithm were as detailed in [1] except that default votes were not used. Thus, if a neighbour has not seen a film then that neighbour is ignored when making a prediction for that film. The Artificial Immune System parameters were set to 'good' values (as observed in the previous paper): thus stimulation rate was set to 0.3 and suppression rate to 0.2. As reported previously, the prediction performance (mean absolute error) was not significantly different between the two algorithms, but recommendation (Kendall's Tau) was significantly better for the Artificial Immune System recommender (as before, a Wilcoxon matched pairs signed rank test was used to assess significance).

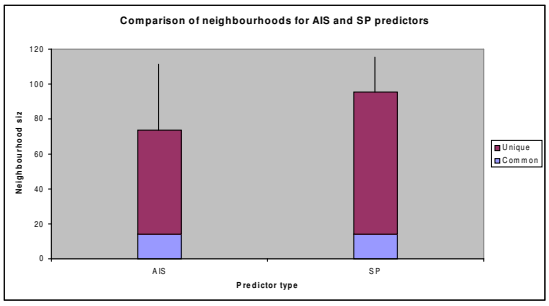


Figure 3: Comparison of Artificial Immune System and Simple Pearson neighbourhoods. The total size of each bar represents the total size of the neighbourhoods produced by each predictor (averaged over 100 predictions; bar shows standard deviation). The lower part of each bar shows the average number of common neighbours (i.e. appearing in both neighbourhoods). The remainder of the bar is composed of unique neighbours – that is, neighbours who appeared in one neighbourhood but not the other.

The first thing to observe is that the neighbourhoods produced by each algorithm are different. As implied from the above, Simple Pearson tended to produce large neighbourhoods (average 95.4 as opposed to 73.8 using the Artificial Immune System) and Figure 3 shows that the composition of these neighbourhoods is different. In particular, it does not seem that the Artificial Immune System neighbourhoods are merely subsets of the Simple Pearson neighbourhoods. In fact, the vast majority of neighbours are 'unique' – that is, chosen by one algorithm but not the other

Is it the neighbourhoods that make the difference to prediction and recommendation performance? Figure 4 shows Artificial Immune System and Simple Pearson performance on both neighbourhoods. For this experiment, we recorded the neighbourhoods found by both the Artificial Immune System and Simple Pearson algorithms.

We then reran the predictions, with everything the same except that this time we forced the Artificial Immune System and Simple Pearson algorithms to use our 'fixed' neighbourhoods. We can see that for prediction, changing the neighbourhood (or indeed algorithm) did not seem to make any significant difference (Table 1 has the details of the statistical tests). However, for recommendation, although the means are very similar (Fig 4), the Artificial Immune System neighbourhood usually produced better recommendations than the Simple Pearson neighbourhood (Table 1b). In fact, the neighbourhood effect seems to dominate, since given the Artificial Immune System neighbourhood, the Simple Pearson algorithm appears to do significantly better than the Artificial Immune System algorithm for recommendation. There is one exception to this trend, where the Artificial Immune System algorithm does not do significantly better for either neighbourhood. In addition, the Artificial Immune System algorithm does better on the Simple Pearson neighbourhood than the Simple Pearson algorithm does, indicating that the neighbour weightings, as well as the neighbours themselves, also contribute to the recommendation quality.

We ran these experiments using default votes (neighbours who had not voted on a film were assumed to give the film a slightly negative rating) and obtained similar results.

It is worth pointing out at this stage that these results should not be taken to be exhaustive, merely indicative. Indeed, we would not want to draw any firm conclusions based on only 100 predictions. This point will be returned to in the discussion. Nevertheless, the results obtained so far seemed to indicate that it was worth investigating the contribution of neighbourhood composition to recommendation performance.

Fig 4a **Fig 4b**

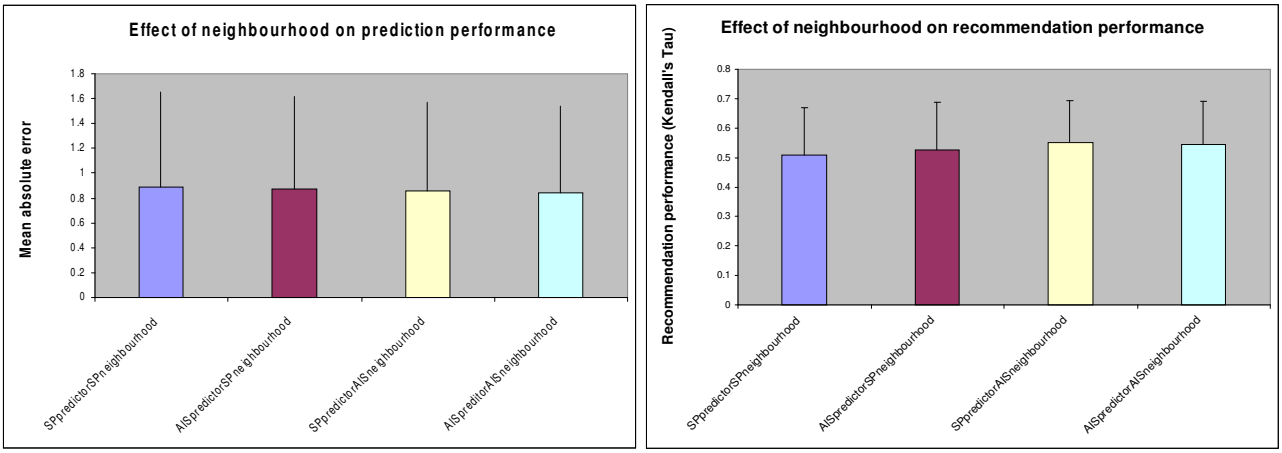


Figure 4: Effect of neighbourhood composition for Artificial Immune System and Simple Pearson algorithms. See text for details on fixing the neighbourhoods. Fig 4a shows prediction performance (measured as mean absolute error averaged over 100 predictions) for each algorithm and each neighbourhood. Fig 4b shows recommendation performance deviation. (measured as Kendall's Tau averaged over 100 predictions) for each algorithm and each neighbourhood. Bars show standard deviation.

Table 1: Analysis of differences between neighbourhoods and algorithms for both prediction (1a) and recommendation (1b). In each case, the Wilcoxon significance test was applied to the results obtained from each pair of regimes. Regimes that are significantly better are shown in **bold** (there were no significant differences found for prediction). [AIS = Artificial Immune System; SP = Simple Pearson]

Table 1a

1st Predictor	1st neighbourhood	2nd Predictor	2nd neighbourhood	Median 1	Median 2	Number of (unequal) predictions compared	1st regime better (sum of ranks)	2nd regime better (sum of ranks)	Significance (upper bound)
SP	SP	AIS	SP	0.682	0.697	97	2212	2541	0.5551
SP	SP	SP	AIS	0.682	0.658	97	2163	2590	0.4434
SP	SP	AIS	AIS	0.682	0.652	97	2176	2577	0.4717
AIS	SP	SP	AIS	0.697	0.658	97	2256	2497	0.6659
AIS	SP	AIS	AIS	0.697	0.652	97	2258	2495	0.6711
SP	AIS	AIS	AIS	0.658	0.652	84	1706	1864	0.7263

Table 1b

1st Predictor	1st neighbourhood	2nd Predictor	2nd neighbourhood	Median 1	Median 2	Number of (unequal) predictions compared	1st regime better (sum of ranks)	2nd regime better (sum of ranks)	Significance (upper bound)
SP	SP	**AIS**	**SP**	0.525	0.557	83	801	2685	1.917e-05
SP	SP	**SP**	**AIS**	0.525	0.549	83	707.50	2778.50	2.617e-06
SP	SP	**AIS**	**AIS**	0.525	0.542	85	930	2725	8.483e-05
AIS	SP	**SP**	**AIS**	0.557	0.549	82	1218.50	2184.50	0.02571
AIS	SP	AIS	AIS	0.557	0.542	80	1426	1814	0.3534
SP	**AIS**	AIS	AIS	0.549	0.542	78	2149	932	0.002459

We looked at a variety of neighbourhood parameters (we might term these community characteristics) across Simple Pearson and Artificial Immune System neighbourhoods. Four characteristics are of particular interest, and each will be discussed in turn. Firstly, it might seem reasonable to assume that performance improves with the number of neighbours in a neighbourhood. However, clearly there is a cost in collecting neighbours (of appropriate quality) together, and thus it will be useful if we can provide good quality recommendations from smaller neighbourhoods.

Another characteristic is the overlap size, which governs the number of recommendations we can assess (An overlap is a test user vote that is also contained in the union of all neighbours' votes). Thirdly, we looked at correlation between each neighbour and the test user. A high correlation shows that neighbours are clustered 'tightly' around the test user, which we might imagine would provide for better recommendations. Fourthly, the idiotypic effect is expected to reduce the inter-neighbour correlations. An obvious intuition might be that such a reduction causes an increase in recommendation quality.

Table 2 shows the difference in these community characteristics across Simple Pearson and Artificial Immune System neighbourhoods. It can be seen that the Artificial Immune System does produce neighbourhoods that are measurably different in character to the Simple Pearson neighbourhoods. In summary, the Artificial Immune System neighbourhoods are smaller, have less overlap, are generally less correlated with the test user and have lower inter-neighbour correlations.

In order to test out which (if any) of these characteristics is crucial, we plotted recommendation performance against each for the Artificial Immune System algorithm. The results seem to show that none of these characteristics *on their own* influences the performance in a clear way. Figure 5 shows scatter plots generated for each characteristic against recommendation quality. Trend lines (based on a power law) have been added to emphasise any underlying data trends.

The first plot suggests that neighbourhood size is not essential in order to obtain high quality recommendations. The second plot, however, does suggest that small overlap sizes might be beneficial for producing good recommendations (regression analysis has not been performed so at this stage this is merely a suggestion). This in some sense is intuitive, as it might be easier to produce higher quality recommendations if there are less of them. However, a balance needs to be struck here; once the overlap size gets too low, the neighbourhood may no longer prove useful to the user.

The third plot shows that, perhaps surprisingly, high correlation between neighbours and the test user may not be essential for high quality recommendations. Finally, the fourth plot would seem to indicate that reduced inter-neighbour correlation is not important in recommendation accuracy, or at least if it is responsible, it is part of a wider effect.

Table 2: Analysis of difference in neighbourhood characteristics between Simple Pearson and Artificial Immune System algorithms. Four characteristics are shown. In each case, the Wilcoxon significance test was applied to the neighbourhoods obtained from the algorithms. In all four cases, the value for the Simple Pearson was significantly higher; this is indicated by **bold** type.

1st Predictor	2nd Predictor	Neighbourhood characteristic tested	Mean 1	Mean 2	Number of (unequal) neighbourhoods compared	1st neighourhood has higher value (sum of ranks)	2nd neighourhood has higher value (sum of ranks)	Significance (upper bound)
Simple Pearson	Artificial Immune System	Neighbours	95.40	73.75	97	4602	151	1.196e-15
Simple Pearson	Artificial Immune System	Overlap	47.46	46.39	26	334.50	16.50	5.686e-05
Simple Pearson	Artificial Immune System	Correlation	0.12	0.10	79	2566	594	1.465e-06
Simple Pearson	Artificial Immune System	Neighbour correlation	0.15	0.04	83	3477	9	3.572e-15

Fig 5a

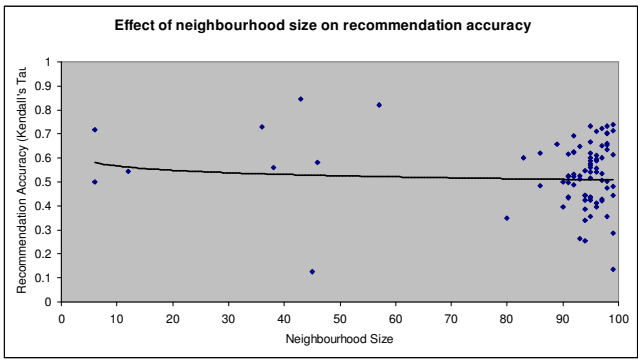


Fig 5b

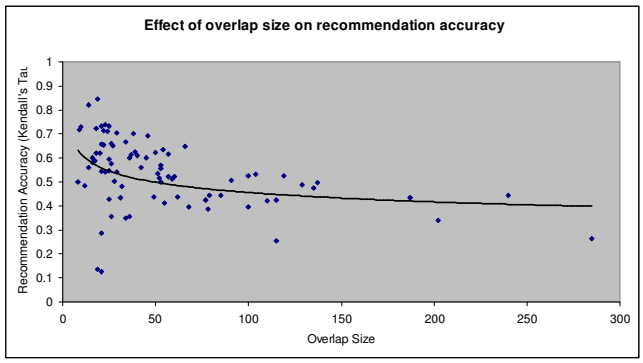


Fig 5c

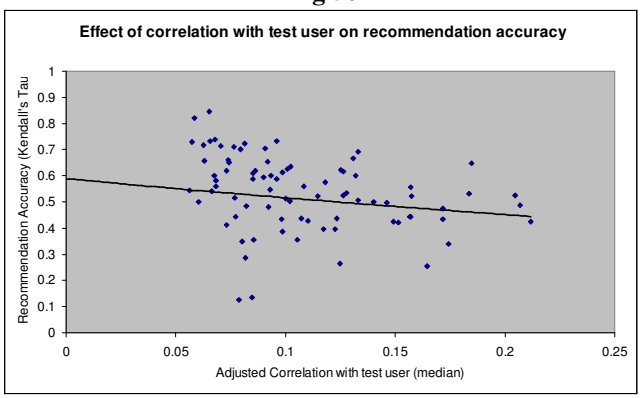


Fig 5d

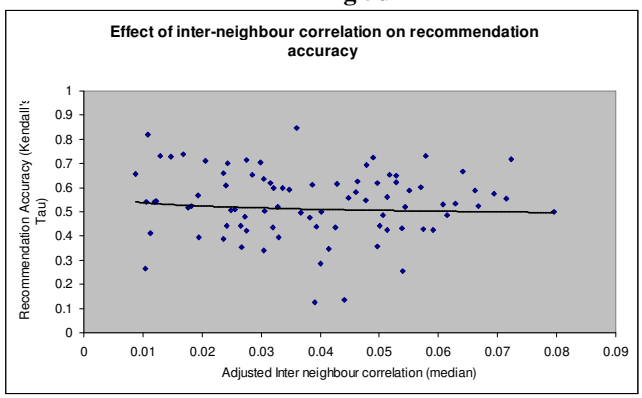


Figure 5. Effect of various neighbourhood measures on Artificial Immune System recommendation performance. In each graph, the measure is shown on the x-axis. The recommendation performance (where available) for each of 100 Artificial Immune System predictions is plotted against this neighbourhood measure. Trend lines are added to indicate the underlying data trend (if any).

5 DISCUSSION AND CONCLUSIONS

As mentioned previously, it is not claimed that these results are conclusive. Indeed, much more data is required before any firm conclusions can be drawn. In this respect, this paper is very much a work in progress. Nevertheless, the results to date certainly are indicative, and challenge certain assumptions. It is hoped that the presentation of these results will stimulate discussion and interest in the nature of the idiotypic effect.

It does not seem likely that the idiotypic effect can be captured by one particular measurement. Nevertheless, it is likely to be some combination of factors. For example, we have shown that both the neighbourhood choice and the weighting of neighbours within that neighbourhood can influence the recommendation performance. Pinning down the effect further has proved to be problematic. Our first intuition – that spreading out neighbours by reducing inter-neighbour correlation improves recommendation – appears to be at best incomplete and at worst incorrect. The mechanisms underlying the effect are clearly subtler than this.

There are of course other community characteristics that we could explore. Some (for example, number of recommendations, overlaps per neighbour, absolute correlation scores) have been examined and shown to be equally inconclusive. Some (for example, number of neighbours voting on each film) remain potential future subjects for investigation.

Other tests (e.g. setting each neighbour's concentration to a random number for immune system predictions, to see whether accurate concentrations are really necessary) might shed further light on the relative importance of each measure. But it is our intuition that such studies might not really get at the nature of the effect, and that larger scale or more sophisticated tests will be needed, coupled with perhaps analytical work, to get at the heart of this intriguing phenomenon.

There are wider implications for such work. The database used for this study [2] is based on real peoples' profiles. Thus, any headway made into improving neighbourhoods by the idiotypic effect can have real benefit for other recommenders – and indeed any community based application.

References

[1] Cayzer S, Aickelin U, A Recommender System based on the Immune Network, Proceedings of the 2002 Congress on Evolutionary Computation, 2002.

[2] Compaq Systems Research Centre. EachMovie collaborative filtering data set, http://www.research.compaq.com/SRC/eachmovie/.

[3] Farmer JD, Packard NH and Perelson AS, The immune system, adaptation, and machine learning Physica, vol. 22, pp. 187-204, 1986.

[4] Gokhale A, Improvements to Collaborative Filtering Algorithms 1999. Worcester Polytechnic Institute. http://www.cs.wpi.edu/~claypool/ms /cf-improve/.

[5] Goldsby R, Kindt T, Osborne B, Kuby Immunology, Fourth Edition, W H Freeman, 2000.

[6] Jerne NK, Towards a network theory of the immune system Annals of Immunology, vol. 125, no. C, pp. 373-389, 1973.

[7] Perelson AS and Weisbuch G, Immunology for physicists Reviews of Modern Physics, vol. 69, pp. 1219-1267, 1997.

An Artificial Immune System as a Recommender for Web Sites

Tom Morrison
University of the West of England
Frenchay Campus Bristol
BS16 1QY
tom.morrison@uwe.ac.uk

Uwe Aickelin
Department of Computing
University of Bradford
BD7 1DP
u.aickelin@bradford.ac.uk

Abstract

Artificial Immune Systems have been used successfully to build recommender systems for film databases. In this research, an attempt is made to extend this idea to web site recommendation. A collection of more than 1000 individuals' web profiles (alternatively called preferences / favourites / bookmarks file) will be used. URLs will be classified using the DMOZ (Directory Mozilla) database of the Open Directory Project as our ontology. This will then be used as the data for the Artificial Immune Systems rather than the actual addresses. The first attempt will involve using a simple classification code number coupled with the number of pages within that classification code. However, this implementation does not make use of the hierarchical tree-like structure of DMOZ. Consideration will then be given to the construction of a similarity measure for web profiles that makes use of this hierarchical information to build a better-informed Artificial Immune System.

1 INTRODUCTION

This research is concerned with using Artificial Immune Systems as a recommender of web sites for new database members. Thus, a new member of the database system would be able to export their bookmark / favourites file and receive a small number of recommendations of web site addresses (URLs or Uniform Resource Locators). Unlike a search engine that will only return specific items a user searches for, our recommender system should be capable of providing the user with surprising items of interest.

Artificial Immune Systems are adaptive search algorithms based on the biological immune system with the central task of pattern matching between antigens and antibodies. Thus in our opinion, they are particularly well suited to data-mining tasks that involve sifting through large databases and finding matches to other items. This has been confirmed in recent research by Cayzer and Aickelin [5] who used Artificial Immune Systems to recommend films to new members of a database based on their rating of at least five films.

As in the research by Cayzer and Aickelin, the type of Artificial Immune System developed here will be based on Jerne's idiotypic network ideas [13]. Hence, we will build an Artificial Immune System that will find a group of users in the database who are similar to the target user in their web site preferences. At the same time, the idiotypic effects will ensure that this group is as diverse as possible. Thus, we will have created an ideal base for predicting and recommending web sites. To do this successfully two steps are necessary: building a database that models individuals' web profiles using a suitable ontology, and constructing a suitable measure of how similar two web profiles are.

The remainder of this paper is organised as follows: In the next section, a very brief overview of the immune system is given with particular emphasis on those features that we intend to exploit here. Section 3 will summarise the research into film prediction and explain differences and similarities to this piece of research. The following section describes the data and ontology used and gives further details about the task of web site recommendation. Section 5 presents a description of the intended Artificial Immune System with an emphasis on the discussion of a suitable similarity measure. The paper is concluded with a summary.

2 THE IMMUNE SYSTEM

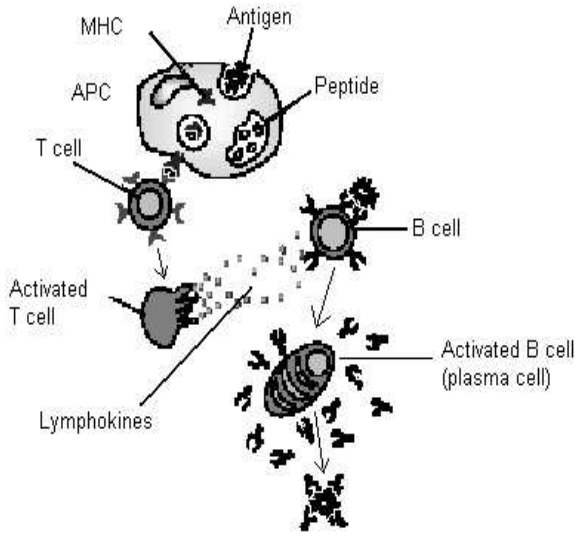
The human body is protected against foreign invaders by a multi-layered immune system. The immune system is composed of physical barriers such as the skin and respiratory system; physiological barriers such as destructive enzymes and stomach acids; and the immune system, which has two complementary parts, the innate and adaptive immune systems. The innate immune system is an unchanging mechanism that detects and destroys certain invading organisms, whilst the adaptive immune system responds to previously unmet foreign cells and builds a response to them that can remain in the body over time.

The immune system is composed of a number of different agents performing different functions at a number of different locations in the body. The precise

interaction of these agents is still a topic for debate [10]. In order to present the important aspects of the system from a mathematical viewpoint it is necessary to simplify and present a selective description.

The immune system's job is to detect antigens, which are foreign molecules from a bacterium or similar invader. The innate immune system helps in the detection process but the main response is through the adaptive immune system. Two of the most important cells in this process are white blood cells, called T cells, and B cells. Both of these originate in the bone marrow but T cells pass on to the thymus to develop before, as with B cells, they circulate the body in the blood and lymphatic vessels.

B cells are responsible for the production and secretion of antibodies, which are specific proteins that bind to the antigen. Each B cell can only produce one particular antibody. The antigen is found on the surface of the invading organism and the binding of an antibody to the antigen is a signal to destroy the



invading cell. A diagram from de Castro and Von Zuben [4] of this process is shown in Figure 1.

Figure 1: Some of the processes involved in the adaptive immune system.

Whilst there is more than one mechanism at work (see [8], [10] or [15] for more details), the essential process for the sake of this research is the matching of antigen and antibody leading to increased concentrations of more closely matched antibodies. In particular, two processes, known as the 'clonal selection theory' by Burnet [3] and the 'idiotypic network theory' by Jerne [13] and [14], are important to us.

The former can be explained as follows: When an antibody strongly matches an antigen the corresponding B cell is stimulated to produce clones of itself that then produce more antibodies. This selection of B cells for cloning on the basis of the antibody match is called the 'clonal selection principle' and will result in increasing concentrations of that antibody in the body.

However, when the B cells clone themselves they do not do so exactly, but mutate slightly. Similarly, B cells may be stimulated when the antibody-antigen match is not perfect. By allowing mutation, the match could become better. However, a number of poorer matches will also be created, and furthermore, some of the newly produced antibodies could even be harmful to our own cells. Such cells will die out under what is known as the 'negative selection principle' [10].

The mutation, mentioned above, is quite rapid, often as much as de Castro and Von Zuben state in [4] "one mutation per cell division". This allows a very quick response to the antigens. This rapid mutation, known as 'somatic hypermutation' [10], may be linked to the 'fitness' of the antibody. Hence, those B cells producing antibodies that are a good match would be subject to less mutation and vice versa for those that are not such a good match.

The idiotypic network theory, introduced by Jerne in [13] and [14], maintains that interactions in the immune system do not just occur between antibodies and antigens, but that antibodies may interact with each other. Hence, an antibody may be matched by other antibodies, which in turn may be matched by yet other antibodies. This activation can continue to spread through the population. However, this interaction can have positive or negative effects on a particular antibody-producing cell. The idiotypic network has been formalised by a number of theoretical immunologists in [15]. This theory could help explain how the memory of past infections is maintained. Furthermore, it could result in the suppression of similar antibodies thus encouraging diversity in the antibody pool.

This last possibility was used in the research by Cayzer and Aickelin [5] in order to preserve diversity. The Artificial Immune System in their research produced a pool of users who were similar to the new entrant to the database, but dissimilar to each other. Whilst this method produced similar performance in predicting film ratings to a k-nearest neighbour approach, the diversity in the pool of recommenders was found to yield statistically significantly improved recommendations. Given the sparseness of the web site search space it may be that suppression of antibodies on similarity grounds might be unnecessary. This will be investigated.

There are a number of successful Artificial Immune System implementations. However, even in the most complex artificial systems only a fraction of the functionality of the biological immune system is exploited. Typically, the antibody-antigen interaction coupled with somatic hypermutation, form the basis for many Artificial Immune System applications. Examples are Timmis et al [18], who used an Artificial Immune System for clustering multivariate data, and Hajela and Yoo [11], who combined a genetic algorithm and an Artificial Immune System to

optimise the design of a 10 bar truss. The research by Timmis et al also applied the idiotypic network theory and were successful in both classifying data and "generalising to cover a larger region of the input space". However, the article does not comment on the effect of modelling a suppression factor between antibodies. Some of the most promising research to date has been conducted in the area of computer security, for instance by Hofmeyr and Forrest in computer network security [12] and by Kim and Bentley for fraud detection [15] and [16].

3 ARTIFICIAL IMMUNE SYSTEMS AS RECOMMENDERS

Whilst most of the applications described above involve somatic hypermutation, Cayzer and Aickelin [5] had only identical cloning, not mutation, in their algorithm. This was because the potential antibodies were actual users of the film database (EachMovie database provided by the Compaq Research Centre [6]). There the task was to find users that were similar to new entrants to the database. Somatic hypermutation was not used, since it is not immediately obvious how to mutate users sensibly such that these artificial entities still represent plausible profiles.

For the same reasons, cloning in our intended Artificial Immune System will make exact copies, too. Future work might include making inexact copies to create novel profiles once appropriate rules for doing so have been established. This could be particularly beneficial when data gathering is expensive or data is otherwise sparse, perhaps due to its sensitive nature, leading to few users being willing to share their information with others.

The main loop of the recommender algorithm is shown in Figure 2 below and is the core of our Artificial Immune System. The aim of this algorithm is to increase the concentrations of those antibodies (database users) that are similar to the antigen (target user). This process is subject to the suppression of similar antibodies following Jerne's idiotypic ideas mentioned above. Thus, over time the Artificial Immune System contains high concentrations of a diverse set of users who have similar film preferences to the target user.

```
Initialise AIS
Encode user for whom to make predictions as antigen Ag
WHILE (AIS not stabilised) & (More data available) DO
  Add next user as an antibody Ab
  Calculate matching score between Ab and Ag
  Calculate matching scores between Ab and antibodies
  WHILE (AIS at full size) & (AIS not stable) DO
    Iterate AIS
  OD
OD
```

Figure 2: Main loop of the Artificial Immune System's (AIS) algorithm for recommendation.

The diagrams in Figure 3 show the idiotypic effect. In the top diagram, antibodies Ab_1 and Ab_3 are very similar and they would have their concentrations reduced in the 'Iterate AIS' stage of the algorithm above. However, in the lower diagram, the four antibodies are well separated from each other as well as being close to the antigen and so would have their concentrations increased.

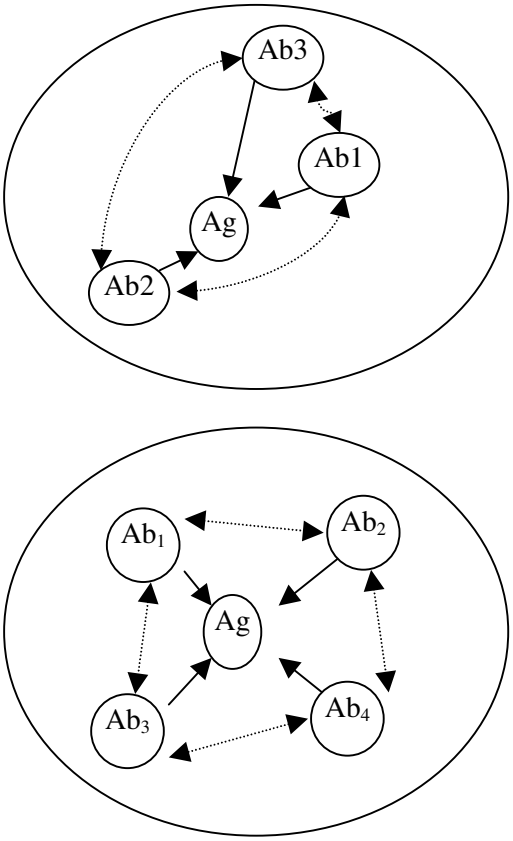


Figure 3: Illustration of the idiotypic effect.

At each iteration of the film recommendation Artificial Immune System the concentration of the antibodies changes according to the formula given below. This will increase the concentration of antibodies that are similar to the antigen and can allow either the stimulation, suppression, or both, of antibody-antibody interactions to have an effect on the antibody concentration. More detailed discussion of these effects on recommendation problems are contained within Cayzer and Aickelin's paper [5].

The following is a formal equation for the idiotypic effect adapted from Equation 3 in Farmer [8]:

$$\frac{dx_i}{dt} = c\left[\begin{pmatrix}antibodies\\recognised\end{pmatrix} - \begin{pmatrix}I\,am\\recognised\end{pmatrix} + \begin{pmatrix}antigens\\recognised\end{pmatrix}\right] - \begin{pmatrix}death\\rate\end{pmatrix}$$

$$= c\left[k_0\sum_{j=1}^{N} m_{ji}x_i x_j - k_1\sum_{j=1}^{N} m_{ij}x_i x_j + k_2\sum_{j=1}^{N} m_{ji}x_i y\right] - k_3 x_i$$

Where:

N is the number of antibodies

x_i is the concentration of antibody i
m_i is the antibody i and the antigen correlation
m_{ij} is the correlation between antibodies i and j
y is the concentration of the antigen
k_1 is suppression, k_2 stimulation and k_3 death rate
k_0 is set to zero in our system, i.e. we do not reward antibody - antibody recognition.

The algorithm is terminated, when the Artificial Immune System is said to have stabilised, i.e. if it has not changed in consistency for more than ten iterations. The concentrations and correlations of the users in the final neighbourhood, i.e. final immune system iteration, are then used to calculate a weighted sum of the ratings of web sites. This would be either a specific unseen web site by the target user in order to predict its ratings, or general top 10 recommendations of new web sites that the target user might enjoy.

4 THE CHALLENGE OF WEB SITE RECOMMENDATION

There are a number of algorithms that recommend items to users. One of the best-known examples is Amazon.com's [1] book recommender based on similar items bought. Generally, these recommenders use what is termed "collaborative filtering" or "social filtering" by Billsus and Pazzani [2]. With the exponential growth of available information on the internet, the need for automated techniques to winnow down the possibilities has also grown but "only a few different algorithms have been proposed in the literature thus far" [2].

Many of the current collaborative filtering techniques use the Pearson correlation coefficient to compare the item ratings of different users. This suffers from several limitations. For example, due to the extremely large amount of information to be rated, two users may only have a very small number of items in common causing the correlation measure to be unduly influenced by those items. Further, there is potentially no difference between the correlation between two users with three items in common and the measure for two users with 30 items in common, in terms of their "influence on the final prediction" [2].

The sparseness of the information space also implies that two users might have no items in common. Can we therefore conclude that they have completely dissimilar tastes, or does the fact that they have not rated particular items imply a similar view of the importance of those items? For these reasons, alternative approaches to both current collaborative filtering algorithms and to the use of the Pearson correlation coefficient should be investigated. More information about traditional and enhanced collaborative filtering is provided by Gokhale [9]. The Artificial Immune System presented here is another example.

In our problem of web site recommendation, the original data consists of sets of web site addresses or URLs taken from bookmark collections such as http://www.cs.ucl.ac.uk/staff/Kim/ComputerImmune. It is extremely unlikely that many people will have many exact addresses in common within their web profiles. Because of this, it is necessary to transform or translate the addresses into a different form. To do this a number of steps are necessary and a widely used web site classification tree ontology will be used called DMOZ [7].

Let us look at the issues involved in the classification of URLs systematically. Typically, an individual web profile in raw form might consist of a list of bookmarks as shown in Figure 4 (in this case taken from the Opera browser – only a small section is shown).

```
#URL
        NAME=ODP - Open Directory Project
        URL=http://dmoz.org/
        CREATED=1017158736
        VISITED=1023875733
#URL
        NAME=Open Directory RDF Dump
        URL=http://dmoz.org/rdf.html
        CREATED=1017159133
        VISITED=1023875759
```

Figure 4: Part of a raw web profile taken from the Opera browser.

This data has to be pre-processed in order to remove unwanted information and superfluous characters. This also includes removing any categories the user might have assigned to some of the bookmarks. Unfortunately, such categorisation of information cannot be kept, as it is arbitrary and individual to the person that owns the bookmarks. For instance, www.bbc.co.uk could be classified under 'media' by one person and under 'news' by another. In addition, misclassifications and duplications might be present in the raw data. Hence, this filtering typically yields a file such as the one partially shown in Figure 5.

```
www.bbc.co.uk/weather/
www.bbc.co.uk/
www.bbc.co.uk/sport/english/football/default.stm
www.guardian.co.uk/
football.guardian.co.uk/
```

Figure 5: Part processed data with superfluous information deleted.

As can be seen from the third line in Figure 5, some of the URLs will have long addresses. Another web profile might contain a very similar address such as *www.bbc.co.uk/sport/english/football/en/default.stm*. If we were to use the raw addresses within the Artificial Immune System, these two would be considered different. However, it is clear that the two users have bookmarked different pages within the

same part of the same site, i.e. 'BBC online - football', and thus have very similar interests.

Therefore, it is still necessary to process the data before it can be used. This presents considerable problems. A program will need to be devised which will truncate the URLs in such a way so that the two addresses discussed above would be considered the same. However, looking again at Figure 4, a simple truncation of the addresses would lead to the first three items occupying the same category. At the same time, it might not lead to the last two being picked together despite the fact that both the addresses refer to pages from the same site. Furthermore, it might not put items 3 and 5 together despite the fact that they are both concerned with football.

To overcome these difficulties, two strategies are used within the DMOZ ontology: Normalisation and reverse partial look-up. First, all URLs undergo a kind of normalisation when pre-formatting the data, as well as when doing look-ups. The protocol and host part are mapped to lowercase characters and host only URLs are always terminated with a "/". During the actual look-up, the category information is gained from DMOZ by employing a reverse truncation search. That is, at first, we try to match the full URL, and then we try to match up to the last "/", then to the last but one "/" etc.

For instance, we would first try to match item three from above by looking for the full URL in DMOZ. If we cannot find that, we would look for www.bbc.co.uk/sport/english/football/; if this fails, we would search for www.bbc.co.uk/sport/english/ etc. Alternatively, we could try to find the closest match in DMOZ defined by the number of consecutive characters that are identical counted from the beginning of the URL.

These normalisation and intelligent matching together should overcome the first problem mentioned above. To overcome problems of misclassification and to have a common standard we decided to use the DMOZ open directory ontology as a classification system [7]. Figure 6 shows part of the structure of this directory.

```
<Topic r:ID="Top/Arts">
<tag catid="2"/>
<d:Title>Arts</d:Title>
<narrow r:resource="Top/Arts/Books"/>
<narrow r:resource="Top/Arts/Music"/>
<narrow r:resource="Top/Arts/Television"/>
[…]
<Topic r:ID="Top/Kids_and_Teens/Pre-School">
<catid>468769</catid>
<link r:resource="http://www.coolplays.com/"/>
<link r:resource="http://kayleigh.tierranet.com/"/>
<link r:resource="http://www.megafile.com.br/"/>
<ExternalPage about="http://www.coolplays.com/">
<d:Title>Coolplay'  s Cool for Kids</d:Title>
<d:Description>Includes animated nursery rhymes, crafts, alphabet and spelling games, and colouring book.
```

Figure 6: Part of the DMOZ open directory structure.

The first half of Figure 6 shows part of the 'Arts' category, which is located immediately below the root of the tree (called Top). Each category has a unique identifier number (2 in this case). This category has a number of sub categories that in turn have several sub categories of their own. In total, there are some 5 million URLs in 428,590 categories spread over 16 levels in the directory. Categories can also be referred to using an address showing the parent categories in a way that preserves the tree structure information. For example, a category address might read '1.3.9' meaning that it is the ninth sub category of category 3, which is the third sub category of category 1.

The second half of Figure 6 shows how URLs are represented in DMOZ and gives an example of a more detailed description of one URL as provided by an anonymous referee. The complete DMOZ database is roughly one GB in size and updated regularly. All specifications in this paper refer to DMOZ as of 1 June 2002. Overall, the version of DMOZ that we use has the following tree structure with deepest branch being 16 levels below the top:

```
  1
  18 /
 621 //
 6675 ///
 30754 ////
 61042 /////
 68901 //////
 101567 ///////
 82802 ////////
 51454 /////////
 20592 //////////
 3467 ///////////
 616 ////////////
  69 /////////////
  8 //////////////
  2 ///////////////
  1 ////////////////
```

Figure 7: Full DMOZ structural tree.

The final stage of processing the data is to turn each of the URLs, shown in Figure 7, into a file containing either the category identification numbers or the category addresses, coupled with the number of items in each category. The choice about which version to use will be discussed in the next section.

There are a number of possible pitfalls with this process. For example, many profiles will contain a set of URLs, which are created by the browser program that they use. Few users are likely to delete all of these links, reasoning that they may be useful at some stage. This may create a situation of artificial similarity between users, which would prevent the Artificial Immune System from functioning effectively.

Secondly, the process of placing URLs into categories is likely to involve some truncation if at first there is no clear category involved. This could lead to several subtly different addresses being classified into the

same category due to the truncation look-up. Depending on whether the truncated sites are from genuinely different URLs or not this could be good or bad. In the first case, the category may appear to be more popular than it should be whereas in the second case the number in the category is a clear indication of interest in that category. Until the data is fully assembled and individual examples are checked, it will not be possible to judge how critical some of these problems will be.

5 BUILDING THE ARTIFICIAL IMMUNE SYSTEM RECOMMENDER

In the film recommender research described in Cayzer and Aickelin [5], each user was coded as a user identification number followed by pairs of film identification numbers with the corresponding rating of the film. The target user became the antigen, whilst the current database members were potential antibodies. In each iteration, antibodies were added to the Artificial Immune System. Those judged to be more similar to the antigen in their film ratings had their concentration increased.

A unique feature of that particular approach was the application of the idiotypic network theory by Jerne [13]. This was implemented such that antibodies that were very similar to each other had their concentration reduced. This has the effect of creating a set of users who are similar to the new user but quite different to each other and thus enhancing the recommendation accuracy of the system. We intend to use the same mechanism for our web site recommender to build an Artificial Immune System as described in section 3.

In order to do this, we also have to decide on the encoding of a user's web profile for which there are two possibilities. In both cases, a user is encoded as a list of category IDs and the number of bookmarks within each category. The difference is in the category IDs; they can be either an integer or a reference to the tree structure. To illustrate the difference, Figure 8 shows the same user's bookmarks for both encodings. The figures in bold indicate how many bookmarks fall into a particular category:

Encoding with the Tree structure:
1.13.12.1.5:**5**;
1.13.12.1.6:**3**;
1.16.3.2.11.5:**1**;
1.18.1.2:**1**;

Encoding with integer category IDs:
22343:**5**;
495771:**3**;
334921:**1**;
3409:**1**;

Figure 8: Integer versus Tree Encoding.

If the second encoding is used together with the number of sites within each category as a rating of the popularity of that category then the problem becomes similar to the film recommendation problem.

However, here we have a considerably sparser search space. In the film database, there were approximately 20,000 entries whereas in the DMOZ directory there are over 400,000 categories. This sparseness may prevent the system from working since many users might have nothing in common, or, at best some categories that are common to the vast majority of the data. Furthermore, many users will have only one entry in a number of categories, leading to increased similarity since the 'rating' of that category will be the same. These problems may prevent an Artificial Immune System based on this encoding being successful in identifying a group of similar users.

There is another problem with using integer category IDs. Because DMOZ is an evolving classification system, new categories are added and removed regularly. This can have the effect that two very similar categories end up with very different integer IDs as these are handed out consecutively. For instance, Star Wars part four might have ID 20,004 when it was classified years ago, but Star Wars part two might end up with ID 420,012 because it has only recently entered the DMOZ system. A similar effect can be seen in Figure 8 for the first two bookmarks. Figure 8 also shows how the tree structure IDs might prevent some of these problems as similar categories still end up near each other in the tree.

The alternative to the integer encoding is to use an encoding that includes the tree structure in the form of a category address. What is required then is a similarity measure that carefully recognises categories that are 'close' within the structure of the tree. For example, it would need to judge the parent / child or the sibling relationship as being more similar than a first cousin or grandparent type relationship. However, constructing such a measure is far from simple. Consider the two trees in Figure 9.

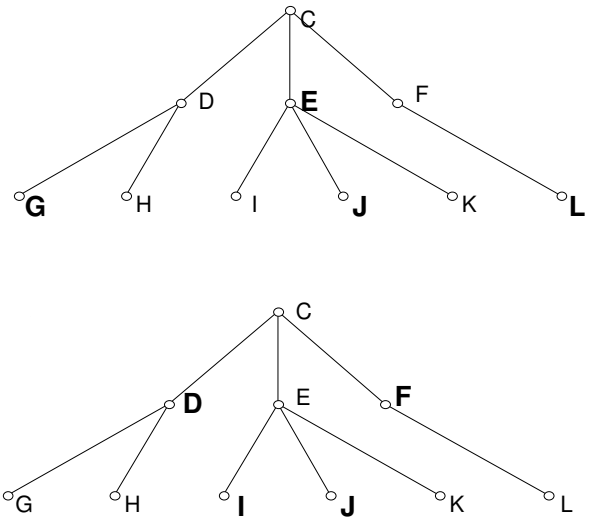


Figure 9: Simple tree structure showing two web profiles.

User 1 has entries at categories G, E, J and L, whilst user 2 has entries at D, I, J and F. Clearly, matches should be scored more highly the lower down the tree they are because this indicates a more precise match. Additionally, 'close' relationships within the tree structure should count more towards the match than ones separated by several 'generations' (to continue the family tree metaphor).

Whilst it is easy to see that these users should have their similarity measure increased, since both have an entry in category J, a question remains what to do with J afterwards. Should this match be discarded once it has been counted by the measure or should the entries at I and J for user 2 be counted as two entries at the parent branch (E) for comparison with user 1? The danger with discarding matches once counted is that two users might have 'perfect' matches for all of the 10 categories that the first user has in their profile, whilst the second user has another 100 entries.

However, if one does not discard categories that have already been matched with another category then it is possible that one quite high level category might be 'matched' with all the different entries at sub-categories for another user. This might not matter since the 'strength' of the match would have been reduced by the generational distance and the weakness of the high-level category's contribution.

6 SIMILARITY MEASURES

Let us now construct a suitable similarity measure for the Artificial Immune System that will produce a value on a 0–1 scale with answers closer to 1 indicating a closer match. Following the discussion in the previous section, the measure will be built according to the following five principles.

1. Matching at categories lower down the tree structure should contribute more to the measure than matching higher up.
2. Matches at the top level of the tree (i.e. the 'Top' category in the DMOZ database should have a contribution of zero.
3. Matching contribution should be reduced for 'imperfect matches' i.e. those not in exactly the same category. The reduction in contribution should be proportional to the generational distance (i.e. a grandparent child relationship has a generational distance of two.)
4. The matching metric should be scaled (averaged) so that it ranges from 0 to 1.
5. The matching metric should take into account all possible matches between the entries in each web profile, i.e. if there are 10 entries in 1 and 20 in the other then all $10 \times 20 = 200$ potential matches should contribute to the measure.

Suppose that we wish to calculate the matching coefficient for the category addresses 1.3.1.1 and 1.3 in the sample tree diagram in Figure 10 below. We need to define an 'edge distance' as the number of 'steps' apart any two addresses are. For example, 1.1 and 1.1.2.2.1 have an edge distance of three, as do 1.2.2.2 and 1.2.1. This equates the relationship between grandparent and grandchild as the same strength as that between siblings.

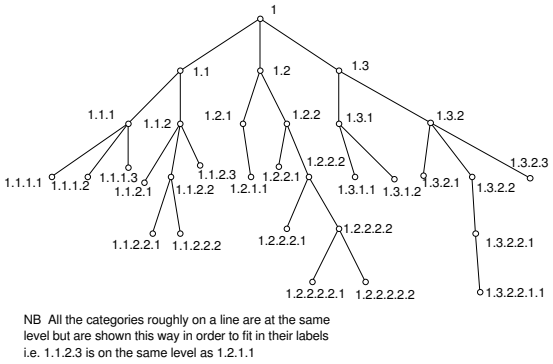


Figure 10: Sample Tree diagram.

By staged truncation of the longer category address (CA) until they are the same we obtain a match at CA 1.3 with two numbers (edge distances) discarded (but counted). This match would have a strength determined by the category level (level 2) of the matching CA, and by the edge distance (ED).

How should the edge distance affect the value of the overall match? One possibility would be to use 1 / ED as this would be a smaller value as the ED increases. However, this would not work when the CA match perfectly as we would be dividing by zero. Therefore using 1 / (ED + 1) is better.

How should the depth of the matching level affect the value of the overall match? It seems useful to make the level number the same as the number of integers in the CA. In the example above, there are six levels. However, the tree is not of uniform depth. In principle, matches at lower levels should score higher since they show a more precise agreement in the topic matter. However, does this mean that a perfect match at the bottom of one set of branches (e.g. 1.1.2.2.2) should score less highly than a perfect match at the bottom of another lower set, say 1.3.2.2.1.1? The DMOZ database is a human classification of human knowledge. To some extent, the classifications are arbitrary because they are the result of pragmatic as well as epistemological considerations. Therefore, it seems incorrect to allow only a perfect match score when it occurs at the lowest level.

In the example above it might be advisable to allow perfect matches to contribute fully at levels 4,5 and 6. Remembering that a match at the top level should count as zero then a formula to give the level effect factor would be (L - 1) / (4 - 1) i.e. level 4 would have a value of 1, level 3 a value of (2/3), level 2 (1/3), whilst the top level would have a value of zero. However, this would not work for values of L greater than 4. To solve this we could use a value of 1 in

those cases. Thus, the general matching formula becomes min{1, (L-1) / (ML-1)} where ML stands for the level at which the maximum contribution starts. In the case of DMOZ, a reasonable choice for the cut-off point might be level 8 based on the structure in Figure 7.

A disadvantage of the measure just described is the inherent simplifications of using a cut-off point after which all matches are equally 'perfect'. The smaller the cut-off value, the more inaccurate result will become. However, if set too large then some branches of the tree might be too shallow to ever achieve a perfect match. It is furthermore questionable whether a linear measure is appropriate. Hence, we propose the following alternative. The matching scores monotonically increasing from level 1 to 16 (in DMOZ's case) but get close to 1 relatively quickly, say at level 8, and then approaches 1 asymptotically as shown in the figure 11.

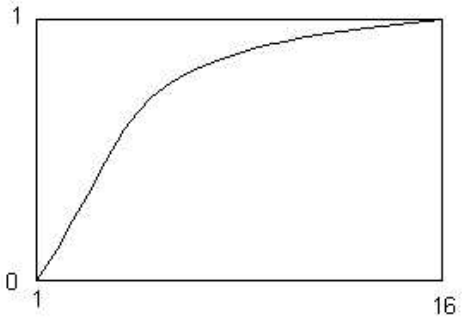


Figure 11: Shape of proposed matching function.

The following equation describes such a function. Let

webprofile1 contain ca_i (i = 1...n) category addresses

webprofile2 contain ca_j (j = 1…m) category addresses

$ed_{i,j}$ be the edge distance from ca_i to ca_j

$l_{i,j}$ be the matching level for ca_i and ca_j

Proposed matching function: $-\frac{l_{i,j}^2 - 33l_{i,j} + 32}{240}$

This measure still agrees with the principle that matches at lower levels should score higher but does not unduly penalise the branches that do not go down to the full 16 levels. Assuming we sum the contributions of all the potential matches the total would have to be divided by the total number of matches to transform the metric to a 0 - 1 scale. Hence, the similarity measure *s* becomes:

$$s = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\frac{1}{ed_{i,j}} \times -\left(\frac{l_{i,j}^2 - 33l_{i,j} + 32}{240}\right)\right)}{n \times m}$$

One further factor should be considered when calculating the match between two web profiles. It is the validity of the match if the web profiles have very different numbers of URLs within them (which we will call the disparity correction factor).

If one web profile has only 10 items whilst the other has 100, then a match from these two people would seem to be less valid than one based on web profiles containing 50 and 60 items. This is because in the first case the 10 entries from the first profile have been used proportionately more in calculating the match. Assuming that web profile 1 (*n* entries) is smaller than web profile 2 (*m* entries) then finding the fraction *n / m* would give a higher result to those pairs of profiles which have similar numbers of entries (see column 3 in Figure 12).

However, it would also give a perfect score to two profiles with a very small number of URLs, say 2 URLs each. Clearly, the measure should 'reward' web profiles that have a larger number of entries. One way to do this would be to include the sum of the number of entries. However, some profiles contain a very large number of entries. Analysis of the data shows that users with more than 100 bookmarks are likely to be outliers. Hence, in order to produce a measure in a range from 0 to 1, profiles with more than 100 entries are counted as though they have 100 entries. Column 4 in Figure 12 shows the calculation of such a measure under the assumptions above.

The fifth column in Figure 12 contains the proposed disparity factor. However, if the raw values in column 5 were used the correction effect would probably be stronger than the original matching score. Therefore a scaling parameter *a* is introduced to reduce the range of the disparity factor. This parameter determines the lowest value in the range (*a*, 1) which the disparity factor can take.

n	m	n/m	(n+m)/200	n/m*(n+m)/200	a+(1-a)*n/m*(n+m)/200
100	100	1.00	1.00	1.00	1.00
80	100	0.80	0.90	0.72	0.89
60	100	0.60	0.80	0.48	0.79
40	100	0.40	0.70	0.28	0.71
20	100	0.20	0.60	0.12	0.65
80	80	1.00	0.80	0.80	0.92
60	80	0.75	0.70	0.53	0.81
40	80	0.50	0.60	0.30	0.72
20	80	0.25	0.50	0.13	0.65
60	60	1.00	0.60	0.60	0.84
40	60	0.67	0.50	0.33	0.73
20	60	0.33	0.40	0.13	0.65
40	40	1.00	0.40	0.40	0.76
20	40	0.50	0.30	0.15	0.66
20	20	1.00	0.20	0.20	0.68
10	20	0.50	0.15	0.08	0.63
10	10	1.00	0.10	0.10	0.64
5	100	0.05	0.53	0.03	0.61
1	100	0.01	0.51	0.01	0.60

Figure 12: Disparity correction using a disparity scaling factor of *a = 0.6*.

Using the same notation as before, with *a* being the scaling parameter for the disparity correction factor the final similarity measure becomes:

$$s = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\frac{1}{ed_{i,j}} \times -\left(\frac{l_{i,j}^2 - 33l_{i,j} + 32}{240}\right) \times (\text{vote } i + \text{vote } j)\right)}{\left(\sum_{i=1}^{n} \text{vote } i + \sum_{j=1}^{m} \text{vote } j\right)} \times \left(a + (1-a)\frac{n(n+m)}{200m}\right)$$

7 CONCLUSIONS

There are a number of steps in the process of preparing the database for use in the Artificial Immune System. These may have an effect on the performance of the system. It will not be possible to tell how critical these issues are until the project is near completion. Having constructed the web profile database the choice of encoding must be made. Again, this could have a critical effect on the success of the Artificial Immune System. It is clear that the construction of a similarity measure that will allow the use of the tree structure is not a trivial task. It may be that this is not necessary and exploration of the potential of the first encoding will be undertaken first since there is already a successful precedent in this case. However, the sparseness of the data set may prevent this, and the creation of a tree comparison similarity measure is an interesting challenge.

To conclude, we believe that with the correct matching metric an idiotypic network based Artificial Immune System should be well suited to supplying interesting yet surprising URLs based on a user's bookmarks. Preliminary results show that with the aid of DMOZ we can map between 60% and 80% of users' bookmarks to votes for suitable categories. We feel confident that this gives us a strong basis for an Artificial Immune System recommender and subsequent result will be published in due course.

Acknowledgements

The authors would like to thank the many volunteers donating their bookmarks and David Banks for his help with the DMOZ system.

References

[1] Amazon.com, http://www.amazon.com.

[2] Billsus, D. and Pazzani, M. (1998). "Learning Collaborative Information Filters" In Shavlik, J., ed., Machine Learning: Proceedings of the Fifteenth International Conference, Morgan Kaufmann Publishers, San Francisco, CA.

[3] Burnet, F. M. (1959) The Clonal Selection Theory of Acquired Immunity. Cambridge University Press, Cambridge.

[4] De Castro, L. N. & Von Zuben, F. J. (1999), Artificial Immune Systems: Part I – Basic Theory and Applications, Technical Report – RT DCA 01/99, FEEC/UNICAMP, Brazil.

[5] Cayzer, S. & Aickelin, U. (2001). A recommender system based on the immune network. Proceedings of CEC 2002.

[6] Compaq Systems Research Centre. EachMovie collaborative filtering data set, http:// www.research.compaq.com/SRC/eachmovie/.

[7] DMOZ ontology, http://dmoz.org/.

[8] Farmer JD, Packard NH and Perelson AS, The immune system, adaptation, and machine learning Physica D, vol. 22, pp. 187-204, 1986.

[9] Gokhale A, Improvements to Collaborative Filtering Algorithms (1999). Worcester Polytechnic Institute. http://www.cs.wpi.edu/ ~claypool/ms/cf-improve/.

[10] Goldsby R, Kindt T, Osborne B (2000), Kuby Immunology, Fourth Edition, W H Freeman.

[11] P. Hajela and J. Yoo (1999), Immune Network Modelling in Design Optimization, New Methods in Optimisation, Editors: (book-chapter) D. Corne, M. Dorigo and F. Glover, McGraw-Hill, pp. 203-216.

[12] Hofmeyr, SA and Forrest, S. (2000). Architecture for an Artificial Immune System. Evolutionary Computation 7, pp 45-68.

[13] Jerne NK (1973), Towards a network theory of the immune system Annals of Immunology, vol. 125, no. C, pp. 373-389.

[14] Jerne, N.K. (1973). The immune system. Scientific American. 229 pp 52-60.

[15] Kim, J. and Bentley, P. J. (2001), Towards an Artificial Immune System for Network Intrusion Detection: An Investigation of Clonal Selection with a Negative Selection Operator, the Congress on Evolutionary Computation (CEC-2001). pp. 1244-1252, 2001.

[16] Kim, J and Bentley, P.J. (2001). An Evaluation of Negative Selection in an Artificial Immune System for Network Intrusion Detection. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001). pp 1330-1337.

[17] Perelson AS and Weisbuch G (1997), Immunology for physicists Reviews of Modern Physics, vol. 69, pp. 1219-1267.

[18] Timmis, J., Neal, M. and Hunt, J. (2000), An Artificial Immune System for Data Analysis. Biosystems 55 pp 143-150.

Artificial Immune Recognition System (AIRS): Revisions and Refinements

Andrew Watkins

Computing Laboratory
University of Kent at Canterbury, UK
and
Department of Computer Science
Mississippi State University, USA.
abw5@ukc.ac.uk

Jon Timmis

Computing Laboratory
University of Kent at Canterbury, UK
J.Timmis@ukc.ac.uk

Abstract

This paper revisits the Artificial Immmune Recognition System (AIRS) that has been developed as an immune-inspired supervised learning algorithm. Certain unnecessary complications of the original algorithm are discussed and means of overcomming these complexities are proposed. Experimental evidence is presented to support these revisions which do not sacrifice the accuracy of the original algorihtm but, rather, maintain accuracy whilst increasing the simplicity and data reduction capabilities of AIRS.

1 INTRODUCTION

Recently, there has been a great deal of interest in the use of the immune system as inspiration for computer science and engineering. These Artificial Immune Systems (AIS) seem to have great potential, which is as yet unrealized. An intuitive application of AIS is in the area of computer security, network intrusion detection (Forrest, Perelson et al. 1994), (Hofmeyr and Forrest 2000) and (Kim and Bentley 2001), change detection, and so on. However, AIS are not limited to this field alone. Work has identified that the immune system contains certain properties that may be useful to create learning algorithms for computer science through the exploitation of the natural learning mechanisms contained within the immune system (Bersini and Varela 1990). However, the focus of current AIS research seems to have been on the development of unsupervised learning algorithms (De Castro and Von Zuben 2000b) and (Timmis and Neal 2001) rather than the supervised or reinforcement kind. An exception to this is work in (Carter 2000). Recent work in (Watkins 2001) explored the possibility of utilizing the immune system as inspiration for the creation of a supervised learning technique. By extracting useful metaphors from the immune system and building on previous immune inspired unsupervised learning algorithms, a classifier was constructed that seems to perform reasonably well on various classification and machine learning problems (Watkins and Boggess 2002a).

This paper presents a further investigation into the work of (Watkins 2001) and suggests improvements to the algorithm that are capable of maintaining classification accuracy, whilst improving performance in terms of computational costs and an increase in the data reduction capabilities of the algorithm. This paper outlines the previous work undertaken in (Watkins 2001), suggests improvements to the algorithms and discusses the implications of these new results. Attention is then given to future possibilities with this approach.

2 BACKGROUND RESEARCH ON AIRS

AIRS (Artificial Immune Recognition System) is a novel immune inspired supervised learning algorithm (Watkins 2001). Motivation for this work came from the author's identification of the fact that there was a significant lack of research that explored the use of the immune system metaphor for supervised learning; indeed, the only work identified was that of (Carter 2000). However, it was noted that within the AIS community there had been a number of investigations on exploiting immune mechanisms for unsupervised learning (that is, where the class of data is unknown a-priori) (Timmis, Neal et al. 2000), (Timmis and Neal 2001) and (De Castro and Von Zuben 2000b). Work in (De Castro and Von Zuben 2000a) examined the role of the clonal selection process within the immune system (Burnet 1959) and went on to develop an unsupervised learning known as CLONALG. This work was extended by employing the metaphor of the immune network theory (Jerne 1974) and then applied to data clustering. This led to the development of the aiNet algorithm (De Castro and Von Zuben 2000b). Experimentation with the aiNet algorithm revealed that evolved *artificial immune networks*, when combined with

traditional statistical analysis tools, were very effective at extracting interesting and useful clusters from data sets. aiNet was further extended to multimodal optimization tasks (De Castro and Timmis 2002b). Other work in (Timmis, Neal et al. 2000) also utilized the immune network theory metaphor for unsupervised learning, and then augmented the work with the development of a resource limited artificial immune network (Timmis and Neal 2001), which reported good benchmark results for cluster extraction and exploration with *artificial immune networks*. Indeed, this work has been further extended by (Nasaroui, Gonzalez et al. 2002) with the introduction of fuzzy logic and refinement of various calculations. The work in (Timmis and Neal 2001) was of particular relevance to (Watkins 2001) and the further work described in this paper.

Building on this previous work, in particular the ideas of artificial recognition balls and resource limitation from (Timmis and Neal 2001) and long-lived memory cells from (De Castro and Von Zuben 2000b). AIRS demonstrated itself to be an effective classifier. The rest of this section describes the immune metaphors that have been employed within AIRS, outlines the algorithm and discusses results obtained, before progressing to the following section, which describes augmentations and improvements to AIRS.

2.1 IMMUNE PRINCIPLES EMPLOYED

A little time should be taken to draw attention to the most relevant aspects of immunology that have been utilized as inspiration for this work. A more detailed overview of the immune system and its relationship with computer science and engineering can be found in (De Castro and Timmis 2002a).

Throughout a person's lifetime, the body is exposed to a huge variety of pathogenic (potentially harmful) material. The immune system contains lymphocyte cells known as B- and T-cells, each of which has a unique type of molecular receptor (location in a shape space). Receptors in this shape space allow for the binding of the pathogenic material (antigens), with the higher affinity (complementarity) between the receptor and antigen indicating a stronger bind. Work in (De Castro and Timmis 2002a) adopted the term shape-space to describe the shape of the data being used, and defined a number of affinity measures, such as Euclidean distance, which can be used to determine the interaction between elements in the AIS. Within AIRS (and most AIS techniques) the idea of antigen/antibody binding is employed and is known as *antigenic presentation*. When dealing with learning algorithms, this is used to implement the idea of matching between training data (antigens) and potential solutions (B-Cells). Work in (Timmis and Neal 2001) employed the idea of an *artificial recognition ball (ARB),* which was inspired by work in (Farmer, Packard et al. 1986) describing antigenic interaction within an immune network. Simply put, an ARB can be thought to represent a number of identical B-Cells and is a mechanism employed to reduce duplication and dictate survival within the population.

Once the affinity between a B-Cell and an antigen has been determined, the B-Cell involved transforms into a plasma cell and experiences *clonal expansion*. During the process of clonal expansion, the B-Cell undergoes rapid proliferation (cloning) in proportion to how well it matches the antigen. This response is antigen specific. These clones then go through *affinity maturation*, where some undertake somatic hypermutation (mutation here is inversely proportional to antigenic affinity) and eventually will go through a selection process through which a given cell may become a memory cell. These memory cells are retained to allow for a faster response to the same, or similar, antigen should the host become re-infected This faster response rate is known as the secondary immune response. Within AIRS, the idea of clonal expansion and affinity maturation are employed to encourage the generation of potential memory cells. These memory cells are later used for classification.

Drawing on work from (Timmis and Neal 2001), AIRS utilized the idea of a stimulation level for an ARB, which, again, was derived from the equations for an immune network described in (Farmer, Packard et al. 1986). Although AIRS was inspired by this work on immune networks, it was found that maintaining a network representation—with connections, stimulation, and repression among the ARBs in the system—was not necessary for evolving a useful classifier. In AIRS, ARBs experience a form of clonal expansion after being presented with training data (analogous to antigens); details on this process are provided in section 2.2. However, AIRS did not take into account the affinity proportional mutation. When new ARBs were created, they were subjected to a process of random mutation with a certain probability and were then incorporated into the memory set of cells should their affinity have met certain criteria. Within the AIRS system, ARBs competed for survival based on the idea of a resource limited system (Timmis and Neal 2001). A predefined number of resources existed, for which ARBs competed based on their stimulation level: the higher the stimulation value of an ARB the more resources it could claim. ARBs that could not successfully compete for resources were removed from the system. The term *metadynamics* of the immune system refers to the constant changing of the B-Cell population through cell proliferation and death. This was present in AIRS with the continual production and removal of ARBs from the population. Table 1 summarizes the mapping between the immune system and AIRS.

Table 1: Mapping between the Immune System and AIRS

IMMUNE SYSTEM	AIRS
Antibody	Feature vector
Recognition Ball	Combination of feature vector and vector class
Shape-Space	The possible values of the data vector
Clonal Expansion	Reproduction of ARBs that are well matched with antigens
Antigens	Training data
Affinity Maturation	Random mutation of ARB and removal of lowest stimulated ARBs
Immune Memory	Memory set of mutated ARBs
Metadynamics	Continual removal and creation of ARBs and Memory Cells

2.2 THE AIRS ALGORITHM

The previous section outlined the metaphors that were employed in the development of AIRS. This section now presents the actual algorithm and discusses the results obtained from experimentation. A more detailed description of the algorithm and results can be found in (Watkins 2001).

Within AIRS, each element (ARB) corresponds to a vector of n dimensions and a class to which the data belongs. Additionally, each ARB has an associated stimulation level as defined in equation 1, where x is feature vector of the ARB, s^x is the stimulation of an ARB x, y is the training antigen, and affinity, in the current implementation, is a function that calculates the Euclidean distance:

$$s^x = \begin{cases} 1\text{-affinity}(x,y), & \text{if class of } x \equiv \text{class of } y \\ \text{affinity}(x,y), & \text{otherwise} \end{cases} \quad (1)$$

Notionally, AIRS has four stages to learning: initialization, memory cell identification, resource competition and finally refinement of established memory cells. AIRS is a one-shot learning algorithm; therefore, the process described below is run for each antigenic pattern, one at a time. Each of these processes will be outlined with the algorithm summarized below.

Initialization of the system includes data pre-processing (normalization) and seeding of the system with randomly chosen data vectors. Assuming a normalized input training data set (antigens), data from that set are randomly selected to form the initial ARB population **P** and memory cells **M**. Prior to this selection, an affinity threshold is calculated; this threshold for the current implementation is the average Euclidean distance between each item in the training data set. This is then used to control the quality of the memory cells maintained as classifier cells in the system.

AIRS maintains a population of memory cells **M** for each class of antigen, which, upon termination of the algorithm, should have identified suitable memory cells to provide a generalized representation for each class of antigenic pattern. The first stage of the algorithm is to determine the affinity of memory cells to each antigen of that class. Then the highest affinity cells are selected for cloning to produce a set of ARBs (which will ultimately be used to create an established memory set). The number of clones that are produced is in proportion to the antigenic affinity, i.e., how well they match; the ARBs also undergo a random mutation to introduce diversification.

The next stage is to identify the strongest, based on affinity to the training instance, ARBs; these will be used to create the established memory set used for classification. This is achieved via a resource allocation mechanism, taken from (Timmis and Neal 2001), where ARBs are allocated a number of resources based on their normalized stimulation levels. At this point, it is worth noting that the stimulation level of an ARB is calculated not only from the antigenic match, but also the class of the ARB. This, in effect, provides reinforcement for ARBs that are of the same class as the antigenic pattern being learnt and that match the antigenic pattern well, in addition to providing reinforcement for those that do not fall into that class and do not match the pattern well.

Once the stimulation of an ARB has been calculated, the ARB is allowed to produce clones (which undergo mutation). The termination condition is then tested to discover if the ARBs are stimulated enough for training to cease on this antigenic pattern. This is defined by taking the average stimulation for the ARBs of each class, and if each of these averages falls above a pre-defined threshold, training ceases for that pattern. This ARB production is repeated until the stopping criteria are met. Once the criteria have been met, then the candidate memory cell can be selected.

A candidate memory cell is selected from the set of ARBs based on its stimulation level and class, with the most stimulated ARB of the same class as the antigen being selected as the candidate. If this candidate cell has a higher stimulation than any memory cell for that class in the established memory set **M**, then it is added to **M**. Additionally, if the affinity of this candidate memory cell with the previous best memory cell is below the affinity threshold, then this established memory cell is removed from the population and replaced by the newly evolved memory cell, thus achieving population control.

This process is then repeated for all antigenic patterns. Once learning has completed, the set of established memory cells **M** can be used for classification. The algorithm is presented below, in terms of immune processes employed.

1. *Initialization*: Create a random base called the memory pool (**M**) and the ARB pool (**P**).
2. *Antigenic Presentation*: for each antigenic pattern do:

   a) *Clonal Expansion*:

   For each element of **M** determine their affinity to the antigenic pattern, which resides in the same class. Select highest affinity memory cell (*mc*) and clone *mc* in proportion to its antigenic affinity to add to the set of ARBs (**P**)

   b) *Affinity Maturation*:

   Mutate each ARB descendant of this highest affinity *mc*. Place each mutated ARB into **P**.

   c) *Metadynamics of ARBs*:

   Process each ARB through the resource allocation mechanism. This will result in some ARB death, and ultimately controls the population. Calculate the average stimulation for each ARB, and check for termination condition.

   d) *Clonal Expansion and Affinity Maturation*:

   Clone and mutate a randomly selected subset of the ARBs left in **P** based in proportion to their stimulation level.

   e) *Cycle*:

   While the average stimulation value of each ARB class group is less than a given stimulation threshold repeat from step 2.c.

   f) *Metadynamics of Memory Cells*:

   Select the highest affinity ARB of the same class as the antigen from the last antigenic interaction. If the affinity of this ARB with the antigenic pattern is better than that of the previously identified best memory cell *mc* then add the candidate (*mc-candidate*) to memory set **M**. Additionally, if the affinity of *mc* and *mc-candidate* is below the affinity threshold, then remove *mc* from **M**.

3. *Cycle*. Repeat step 2 until all antigenic patterns have been presented.

2.3 RESULTS AND DISCUSSION

AIRS was tested on a number of benchmark data sets in order to assess the classification performance. This section will briefly highlight those results and discuss potential improvements for the algorithm, more details can be found in (Watkins and Boggess 2002a).

Once a set of memory cells has been developed, the resultant cells can be used for classification. This is done through a k-nearest neighbor approach. Experiments were undertaken using a simple linearly separable data set, where classification accuracy of 98% was achieved using a k-value of 3. This seemed to bode well, and further experiments were undertaken using the Fisher Iris data set, Pima diabetes data, Ionosphere data and the Sonar data set, all obtained from the repository at the University of California at Irvine (Blake and Merz 1998). Table 2 shows the performance of AIRS on these data sets, a full comparison table of AIRS and other techniques can be found in (Watkins and Boggess 2002a).

Table 2: AIRS Classification Results on Benchmark Data

IRIS	IONOSPHERE	DIABETES	SONAR
96.7	94.9	74.1	84.0

These results were obtained from averaging multiple runs of AIRS, typically consisting of three, or more, runs and five-way, or greater, cross validation. More specifically, for the Iris data set a five-fold cross validation scheme was employed with each result representing an average of three runs across these five divisions. To remain comparable to other experiments reported in the literature, the division between training and test sets of the Ionosphere data set as detailed in (Blake and Merz 1998) was maintained. However, the results reported here still represent an average of three runs. For the Diabetes data set a ten-fold cross validation scheme was used, again with each of the 10 testing sets being disjoint from the others and results were averaged over three runs across these data sets. Finally, the Sonar data set utilized the thirteen-way cross validation suggested in the literature (Blake and Merz 1998) and was averaged over ten runs to allow for more direct comparisons with other experiments reported in the literature. During the experimentation, it was noted by the authors that varying system parameters such as number of seed cells varied performance on certain data sets, however, varying system resources (i.e., the numbers of resources an ARB could compete for) seemed to have little affect. A comparison was made between the performance of AIRS and other benchmark techniques, where AIRS seemed not to outperform specialist techniques, but on more general purpose algorithms, such as C4.5, it did outperform.

Even though initial results from AIRS did look promising, it can be said there are a number of potential areas for simplification and improvement. There is clearly a need to understand exactly why and how AIRS behaves the way it does. This can be achieved through a rigorous analysis of the algorithm, examining the behavior of the ARB pool and memory set over time. To date, the focus has been primarily on the classification performance of AIRS. Indeed, the final chapter of (Watkins 2001) suggests that an investigation into the resource allocation mechanism would be a useful area of investigation. The majority of AIS techniques use the metaphor of somatic hypermutation or affinity proportional mutation. To date, AIRS does not employ this metaphor but instead uses a naïve random generation of mutations.

The remaining sections of this paper undertake these investigations and present a modified version of AIRS, which is more efficient in terms of ARB production, employs affinity proportional mutation and assess what, if any, difference this has made to the overall algorithm.

3 A MORE EFFICIENT AIRS

Motivated by the observations in (Watkins 2001), current work has focused on refining AIRS. This section details the observations that have been made through a thorough investigation into AIRS and how issues raised through these observations have been overcome.

3.1 OBSERVATIONS

3.1.1 The ARB Pool

A very crude visualization[1] was used to gain a better understanding of the development of the ARB pool. In AIRS there are 2 independent pools of cells, the memory cell pool and the ARB pool. The initial formulation of AIRS uses the ARB pool to evolve a candidate memory cell of the same class as the training antigen, which can potentially enter the memory cell pool. During this evolution, ARBs of a different class than the training antigen were also maintained in the ARB pool. The stimulation of an ARB was based both on affinity to the antigen and class, where highly stimulated ARBs were those of the same class as the antigen and that were "close" to the antigen, or of a different class and "far" from the antigen. However, the visualization revealed that during the process of evolving a candidate memory cell, there seems no need to maintain or evolve ARBs that are a different class than the training antigen. The point of the interaction of the ARB pool with the antigenic material is really only in evolving a good potential memory cell, and this potential memory cell **must** be of the same class as the training antigen. When observing the visualization for a while, it is possible to notice that there is a process of convergence by ARBs of the same class to the training antigen. Naturally, based on the reward scheme, ARBs of a different class are moving further away from the training antigen. However, this process essentially must start over for the introduction of each new antigen, and, therefore, previously existing ARBs are fairly irrelevant. Since there are 2 separate cell pools, with the true memory of the system only being maintained in the Memory Cell pool, maintaining any type of memory in the ARB pool is unnecessary. This change to the algorithm rather than being about resource allocation schemes as initially suggested in (Watkins 2001) is really a simplification to the algorithm, which is seen as a positive step. This simplification affects both memory usage and computational simplification, although this will not be discussed in this paper.

[1] See http://www.cs.ukc.ac.uk/people/rpg/abw5/ARB_hundred.html

3.1.2 Mutation of Cells

Motivated by observing the success of other AIS work, as well as by some of the tendencies discussed in (Watkins 2001) and (Watkins and Boggess 2002b), attention was paid to the way in which mutation occured within AIRS. In these two works, the authors notice that some of the evolved memory cells do not seem as high-quality of classifier cells as some of the others. Additionally, it was observed that there seemed to be some redundancy in the memory cells that were produced. In (De Castro and Von Zuben 2000a) and other AIS work, mutation within an antibody or B-Cell is based on its affinity, with higher affinity cells being mutated less than lower affinity cells. These other AIS works have used this method of somatic hypermutation to a good degree of success. It was thought that embedding some of this approach in AIRS might result in higher quality, less redundant, memory cells. This approach was therefore adopted within AIRS.

3.2 AIRS: WHAT IS NEW?

For the remainder of this section changes that have been made to the AIRS algorithm are described. There then follows empirical results from the new formulation and discuss the implications of these results.

3.2.1 Memory Cell Evolution

In the newly formulated version of AIRS, candidate memory cell evolution is based only on ARBs of the same class as the training antigen. This means that ARBs in the ARB pool are no longer permitted to mutate class. Therefore, the ARB pool will only consist of ARBs that are of the same class as the training antigen. At the end of each antigenic presentation cycle, the pool can be either be cleared out, or the ARBs can stay in the pool. If the pool is not cleared out then it will contain ARBs of all potential classes. The algorithm is only reinforcing the class of the antigenic pattern, and therefore, all ARBs that are in the pool at the end of the antigenic cycle that are not of the same class as the antigenic pattern will be removed through the metadynamic process, as they are no longer rewarded with any resources. This is in contrast to the original formulation of AIRS in which the

allocation of resources, and thus cellular reinforcement, was based on a stimulation value that was calculated as in Equation 1 (section 2.2). In that original version both ARBs "near" the antigen and of the same class as the antigen were rewarded and ARBS "far" from the antigen and of a different class than the antigen were rewarded. Also, ARBs were allowed to mutate their class values (mutate in this case means switching classes). In the newly proposed version of AIRS, only ARBs of the same class are rewarded and mutation of the class value is no longer permitted.

Based on this new formulation, the only user parameter changes that might need to be made is that the stimulation threshold could potentially need to be raised. Recall, that the stimulation threshold was used as a stopping criterion for training the ARB pool on an antigen. In order to stop training on an antigen the average normalized stimulation level had to exceed the stimulation threshold for each class group of ARBs. That is, in a 2-class problem, for example, the average normalized stimulation level of all class 0 ARBs had to be above the stimulation threshold, and the average normalized stimulation level of all class 1 ARBs has to be above the stimulation threshold. It was possible, and frequently the case in fact, that the average normalized stimulation level for the ARBs of the same class as the training antigen reached the stimulation threshold before the average normalized stimulation level of ARBs in different classes from the antigen. What this did, in effect, was allow for the evolution of even higher stimulated ARBs of the same class while they were waiting for the other classes to reach the stimulation threshold. By taking out these extra cycles of evolution through no longer worrying with ARBs of different classes, it is possible that the ARBs will not have converged "as much" as in the previous formulation. This can be overcome by raising the stimulation threshold and thus requiring a greater level of convergence.

3.2.2 Somatic Hypermutation

To explore the role of mutation on the quality of the memory cells evolved, the mutation routine was modified so that the amount of mutation allowed by a given gene in a given cell is dictated by its stimulation value. Specifically, the higher the normalized stimulation value, the smaller the range of mutation allowed. Essentially, the range of mutation for a given gene = 1.0 - the normalized stimulation value of the cell. Mutation is then controlled over this range with the original gene value being placed at the center of the range. This, in a sense, allows for tight exploration of the space around high quality cells, but allows lower quality cells more freedom to explore widely. In this way, both local refinement and diversification through exploration are achieved.

3.3 THE AIRS V2 ALGORITHM

The changes made to the AIRS algorithm are small, but end up having an interesting impact on both the simplicity of implementation and on the quality of results. Section 4 will offer more discussion by way of comparison. For now, the changes to the original AIRS presented in section 2.2 will be discussed. These can be identified as follows:

1. Only the Memory Cell pool is seeded during initialization rather than both the MC pool (**M**) and the ARB pool (**P**). Since we are no longer concerned about maintaining memory or class diversity within **P** it is no longer necessary to initialize **P** from the training data or from examples of multiple classes.
2. During the clonal expansion from the matching memory cell used to populate **P**, the newly created ARBs are no longer allowed to mutate class. Again, maintaining class diversity in **P** is not necessary.
3. Resources are only allocated to ARBs of the same class as the antigen and are allocated in proportion to the inverse of an ARB's affinity to the antigen.
4. During affinity maturation (mutation), a cell's stimulation level is taken into account. Each individual gene is only allowed to change over a finite range. This range is centered with the gene's pre-mutation value and has a width the size of the difference of 1.0 and the cell's stimulation value. In this way the mutated offspring of highly stimulated cells (those whose stimulation value is closer to 1.0) are only allowed to explore a very tight neighborhood around the original cell, while less stimulated cells are allowed a wider range of exploration. (It should be noted that during initialization all gene values are normalized so that the Euclidean distance between any two cells is always within one. During this normalization, the values to transform a given gene to within the range of 0 and 1 are discovered, as well. This allows for this new mutation routine to take place in a normalized space where each gene is in the range of 0 and 1.)
5. The training stopping criterion no longer takes into account the stimulation value of ARBs in all classes, but now only accounts for the stimulation value of the ARBs of the same class as the antigen. In the new formulation of AIRS it is still possible to have ARBs in **P** of different classes if the implementation does not clear the ARB pool after each antigenic pattern. However, this will not affect the stopping criterion since the changes to the algorithm now only require that the average stimulation value of the ARBs of the **same** class as the antigen be above the user-supplied stimulation threshold.

3.4 RESULTS AND DISCUSSION

To allow for comparison between the two versions of the algorithm, the same experiments were performed on the new formulation of AIRS (AIRS2). Section 4 will provide a more thorough comparative discussion, but for now, results of AIRS2 on the four, previously discussed, benchmark sets are presented in Table 3.

Table 3: AIRS2 Classification Results on Benchmark Data

IRIS	IONOSPHERE	DIABETES	SONAR
96.0	95.6	74.2	84.9

These results were obtained by following the same methodology as the original results reported in section 2.3 which is elaborated upon in (Watkins 2001) and (Watkins and Boggess 2002a). Again, we note that these results are competitive with other classification techniques discussed in the literature, such as C4.5, CART, and Multi-Layer Perceptrons.

4 COMPARATIVE ANALYSIS

This section briefly touches on some comparisons between the original version of AIRS presented in discussed in section 2 (AIRS1) and the revisions to this algorithm presented in section 3 (AIRS2). The focus of this discussion will be on two of the more important features of the AIRS algorithms: classification accuracy and data reduction.

4.1 CLASSIFICATION ACCURACY

The success of AIRS1 as a classifier (cf, (Watkins and Boggess 2002a)) makes it important to assess any potential changes to the algorithm in light of test set classification accuracy. To aid in this task, Table 4 presents the best average test set accuracies, along with the standard deviations, achieved by both versions of AIRS on the four benchmark data sets.

Table 4: Comparative Average Test Set Accuracies

	AIRS1: Accuracy	AIRS2: Accuracy
Iris	96.7 (3.1)	96.0 (1.9)
Ionosphere	94.9 (0.8)	95.6 (1.7)
Diabetes	74.1 (4.4)	74.2 (4.4)
Sonar	84.0 (9.6)	84.9 (9.1)

It can be noted that the revisions to AIRS presented in section 3 do not require a sacrifice in classification performance of the system. In fact, for 3 of the 4 data sets we see a slight improvement in the accuracy; however, these differences are not statistically significant. What is important to note is that the changes introduce no fundamental differences in classification accuracy of the system.

4.2 DATA REDUCTION

From the previous subsection it can be seen that the changes introduced to AIRS offer no real difference in classification accuracy, so the question arises: why bother? Why introduce these changes to a perfectly reasonably performing classification algorithm? The answer lies in the data reduction capabilities of AIRS.

In (Watkins 2001) and (Watkins and Boggess 2002b), the authors discuss that aside from competitive accuracies another intriguing feature of the AIRS classification system is its ability to reduce the number of data points needed to characterize a given class of data from the original training data to the evolved set of memory cells. Given the volumes of data involved with many real-world data sets of interest, any technique that can reduce this volume while retaining the salient features of the data set is useful. Additionally, it is this collection of memory cells that are the primary classifying agents in the evolved system. Since classification is, currently, performed in a *k*-nearest neighbor approach, whose classification time is dependent upon the number of data points used for classifying a previously unseen data item, any reduction in the overall number of evolved memory cells is also useful for the algorithm.

Table 5 presents the average size of the evolved set of memory cells and the amount of data reduction this represents in terms of population size and percentage reduction, along with standard deviations, for each version of the algorithm on the four benchmark data sets. The original training set size is also presented for comparison. There are two points of interest:

1. Both versions of the algorithm exhibit data reduction, and
2. AIRS2 tends to exhibit greater data reduction than AIRS1.

Table 5: Comparison of the Average Size of the Evolved Memory Cell Pool

	Training Set Size	AIRS1: Memory Cells	AIRS2: Memory Cells
Iris	120	42.1/65% (3.0)	30.9/74% (4.1)
Ionosphere	200	140.7/30% (8.1)	96.3/52% (5.5)
Diabetes	691	470.4/32% (9.1)	273.4/60% (20.0)
Sonar	192	144.6/25% (3.7)	177.7/7% (4.5)

		(3.7)	(4.5)

This second point is the more important for our current discussion. As mentioned in sections 3.1.2 and 3.2.2, one of the goals of the revision of the AIRS algorithm was to see if employing somatic hypermutation through a method more in keeping with other research in the AIS field would increase the efficiency of the algorithm. The current measure of efficiency under concern is the amount of data needed to represent the original training set to achieve accurate classifications. We can see from Table 5 that, in general, AIRS2 was able to achieve the comparable accuracy presented in section 4.1 with greater efficiency. In fact for some of the data sets, most notably Ionosphere and Diabetes, the degree of data reduction is greatly increased (from 30% to 52% for Ionosphere data and from 32% to 60% for the diabetes data set). Interestingly, for the most difficult classification task, the Sonar data set, the degree of data reduction is not increased. While this was not the general trend on this data set (data not presented), it does possibly point to some limitations in the current version of AIRS. Overall, however, it seems reasonable to claim that the revisions to AIRS provide greater data reduction, and hence greater efficiency, without sacrificing accuracy.

4.3 A WORD ABOUT SIMPLICITY

While the focus has not been on algorithmic complexity analysis of the two versions of AIRS for this current paper, it would be remiss not to make a brief mention concerning the simplifying effects of the revision to AIRS. As mentioned in section 3.1, the reformulation of AIRS was chiefly motivated by some basic observations about the workings of the system. One observation was that the original version of AIRS maintained representation of too many cells for its required task. This led to the elimination of maintaining multiple classes of cells in the ARB pool or of retaining cells in the ARB pool at all. This has the simplifying effect of reducing the memory necessary to run the system successfully. A second observation concerning the quality of the evolved memory cells led to the investigation of the mutation mechanisms employed in the original algorithm. By adopting an approach to mutation proven to be successful in other AIS, it has been possible to increase the quality of the evolved memory cells that is evidenced by the increased data reduction without a decrease in classification accuracy. Both of these overarching changes (ARB pool representation and the mutation mechanisms used) have exhibited a simplifying effect on the classification system as a whole.

5 CONCLUSIONS AND FUTURE WORK

This paper has focused on a supervised learning system based on immunological principles. The Artificial Immune Recognition System (AIRS) introduced in (Watkins 2001) exhibited initial success as a classification algorithm. However, as with any initial system, there were some revisions and refinements that could be made to AIRS that would decrease the complexity of the system. This paper has presented investigations for two of these revisions.

It was shown that the internal data representation of the original version of AIRS was overcomplicated. By simplifying the evolutionary process, it was possible to decrease this complexity whilst still maintaining accuracy. It was also shown that the use of affinity aware mechanisms of somatic hypermutation, as adopted throughout the AIS community, led to higher quality memory cells in AIRS and thus greater data reduction and faster classification of test data items.

Both of these revisions were the result of careful observation of the behavior of the original algorithm. In this respect, it can be said that this paper is also about the importance of taking the steps to investigate the behavior of a system even if it is performing in a successful manner. This paper has demonstrated that such an investigation is fruitful in simplifying the workings without sacrificing the performance of the system.

There are many avenues that can be explored with this work. One is the analogy of this work with reinforcement learning strategies, it could possibly be argued that AIRS is a reinforcement learning algorithm, when one considers certain mechanism within the immune system (Bersini and Varela 1994); this warrants further investigation. Additionally, the role of parallel and distributed processing could be examined, in order to allow for dealing with larger scale problems. Work has already begun on applying AIRS to immunological data, attempting to predict the binding of receptors and in effect trying to solve an immunological problem with an artificial immune system.

References

Bersini, H. and F. J. Varela (1990). Hints for Adaptive Problem Solving Gleaned from Immune Networks. *Parallel Problem Solving from Nature*. pp.343-355

Bersini, H. and F. J. Varela (1994). The Immune Learning Mechanisms: Reinforcement, Recruitment and their Applications. *Computing with Biological Metaphors*. R. Paton, Chapman and Hall**:** 166-192.

Blake, C. L. and C. J. Merz (1998). UCI Repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html

Burnet, F. M. (1959). The Clonal Selection Theory of Immunity, Vanderbilt University Press, Nashville, TN.

Carter, J. H. (2000). "The Immune System as a model for Pattern Recognition and Classification." *Journal of the American medical Informatics Association* **7**(1).

De Castro, L. N. and J. Timmis (2002b). "An Artificial Immune Network for Multimodal Optimisation." *Congress on Evolutionary Computation. Part of the World Congress on Computational Intelligence:* 699-704.

De Castro, L. N. and J. I. Timmis (2002a). Artificial Immune Systems: *A New Computational Intelligence Approach*, Springer-Verlag.

De Castro, L. N. and F. Von Zuben (2000a). "The clonal selection algorithm with engineering applications." *Proceedings of Genetic and Evolutionary Computation* Conference: 36-37.

De Castro, L. N. and F. Von Zuben (2000b). "An Evolutionary Immune Network for Data Clustering." SBRN '00 **1**: 84-89.

Farmer, J. D., N. H. Packard, et al. (1986). "The Immune System, Adaptation, and Machine Learning." *Physica* **22**(D): 187-204.

Forrest, S., A. Perelson, et al. (1994). "Self-Nonself Discrimination in a Computer." *Symposium on Research in Security and Privacy*: 202-212.

Hofmeyr, S. and S. Forrest (2000). "Architecture for an Artificial Immune System." *Evolutionary Computation* **7**(1): 45-68.

Jerne, N. K. (1974). "Towards a Network Theory of the Immune System." *Annals of Immunology* **125C**: 373-389.

Kim, J. and P. Bentley (2001). "Towards an Artificial Immune System for Network Intrusion Detection: An Investigation of Clonal Selection with Negative Selection Operator." *Congress on Evolutionary Computation*: 1244-1252.

Nasaroui, O., F. Gonzalez, et al. (2002). "The Fuzzy Artificial Immune System: Motivations, Basic Concepts and Application to Clustering and Web Profiling." *International Joint Conference on Fuzzy Systems*: 711-717.

Timmis, J. and M. Neal (2001). "A resource limited artificial immune system for data analysis." *Knowledge Based Systems* **14**(3-4): 121-130.

Timmis, J., M. Neal, et al. (2000). "An Artificial Immune System for Data Analysis." *BioSystems* **55**(1/3): 143-150.

Watkins, A. (2001). AIRS: A resource limited artificial immune classifier. Department of Computer Science, Mississippi State University. http://nt.library.msstate.edu/etd/show.asp?etd=etd-11052001-102048

Watkins, A. and L. Boggess (2002a). "A new classifier based on resource limited artificial immune systems." *Proceedings of Congress on Evolutionary Computation. Part of the World Congress on Computational Intelligence*.: 1546-1551.

Watkins, A. and L. Boggess (2002b). "A resource limited artificial immune classifier." *Proceedings of Congress on Evolutionary Computation. Part of the World Congress on Computational Intelligence*: 926-931.

A Model of Gene Library Evolution in the Dynamic Clonal Selection Algorithm

J. Kim
Department of Computer Science
King's College London
Strand
London WC2R 2LS
jungwon@dcs.kcl.ac.uk

P. J. Bentley
Department of Computer Science
University College London
Gower Street
London WC1E 6BT
P.Bentley@cs.ucl.ac.uk

Abstract

The dynamic clonal selection algorithm (dynamiCS) was created to tackle the difficulties of anomaly detection in continuously changing environments (Kim and Bentley, 2002a). This algorithm was extended in a sister paper (Kim and Bentley, 2002b), so that memory detectors that are no longer valid are automatically deleted. Here we describe a further extension to the system: the use of hypermutation on deleted memory detectors to produce, in effect, a "virtual gene library" which seeds the immature detector population.

1 INTRODUCTION

When using an Artificial Immune System (AIS) in a real environment (e.g., monitoring network traffic), normal or *self* behaviours can change after a certain period. In addition, the system may only see a small subset of self antigens at any one time. In order for our AIS to be able to deal with such an environment, a dynamic clonal selection algorithm (DynamiCS) was introduced in previous work (Kim and Bentley, 2002a). The results described there showed that DynamiCS could incrementally learn the globally converged distributions even though only one subset distribution was given at each generation. This feature was achieved by employing three important parameters: *tolerisation period* of an immature detector (*T*), *activation threshold* of a mature detector (*A*) and the *life span* of a mature detector. However, the original DynamiCS could not learn new self-antigens when learned self and non-self behaviours suddenly altered due to legal self change. This resulted in high false positive (FP) rates when new antigens were monitored by DynamiCS, although it produced high true positive (TP) rates.

A sister paper to this describes a further extension of DynamiCS, which reduces FP rates increased by memory detectors (Kim and Bentley 2002b). The extended DynamiCS handles generated memory detectors based on their detection results. The original DynamiCS preserved memory detectors for an infinite lifespan. In contrast, the extended DynamiCS kills memory detectors if they show poor self-tolerance to new antigens (Kim and Bentley 2002b). This extended system was tested to determine whether surviving memory detectors no longer cause seriously high FP error rates or not. From this test, it was analysed to see whether any other problems occur as a consequence of killing memory detectors. The analysis showed that the extended DynamiCS requires a larger amount of co-stimulation if it yielded high TP rates.

This analysis led to the work described in this paper: the addition of hypermutation to the extended DynamiCS, to – in effect – evolve a gene library of the AIS. This additional extension is designed to fine-tune generated memory detectors so that the system obtains higher TP rates without increasing the amount of co-stimulation. Here, the new extension is tested to determine whether it gains high TP rates without increasing the amount of co-stimulation as the result of gene library evolution. The test results are then analysed to see how hypermutation leads to such a gene library evolution effect, and thus whether it improves the overall system performance. Finally, the novel features of DynamiCS studied in this work are discussed in accordance with a comparison to the most similar AIS developed by (Hofmeyr, 1999; Hofmeyr and Forest, 2000).

2 DYNAMIC CLONAL SELECTION (DynamiCS) ALGORITHM

The new AIS introduced in previous work (Kim and Bentley, 2002a) follows the basic concept of the AIS proposed by Hofmeyr (1999). The adaptability of Hofmeyr's AIS was achieved via co-ordinated dynamics of three different detector populations: immature, mature, and memory detector populations. In order to fully comprehend the co-ordinated dynamics of these three detector populations in terms of AIS adaptability, we introduced an artificial immune algorithm, called the *dynamic clonal selection algorithm* (DynamiCS). Although Hofmeyr proposed various new features in order to effect great adaptability and distributed detection, DynamiCS attempts to distill only the crucial components that yield adaptability to the system (and reduce the number of system parameters to ensure the algorithm is usable). The following pseudo code provides an overview of the extended DynamiCS.

```
Initialise Dynamic Clonal Selection Algorithm
Create an initial immature detector population with random detectors;

Generation_Number = 1;
Do
{ If (Generation_Number = N) then Select a new antigen cluster.
 Select 80% of self and non-self antigens from chosen antigen cluster;

 Reset Parameters
     Generation_Number++;   Memory Detector Age++;
     Mature Detector Age++;   Immature Detector Age++;

 Monitor Antigens
 {   Monitor Antigens by Memory Detectors
      Co-stimulation: does the memory detector detect a non-self
      antigen or does it detect a self antigen?
      Kill memory detectors that detect self antigens.

    Monitor Antigens by Mature Detectors
       Check whether any mature detector detects any non-self antigen;
       Check whether any mature detector detects any self antigen;
       Create new memory detectors;
       Old mature detectors are killed;

     Monitor Antigens by Immature Detectors
        Check whether any immature detector detects any self antigen;
        Delete any immature detector matching any self antigen;
        Create new mature detectors;
 }

 If (immature detector population size +
     mature detector population size
    < non-memory detector pop size)
 {  Do
    {  Generate a random detector;
        Add a random detector to an immature detector population;
    }  Until  (immature detector population size +
                mature detector population size =
                non-memory detector pop size);
 }
 } While (generation Number < max Generation)
```

Full details of this algorithm are given in (Kim and Bentley, 2002a and b).

All experiments used the Wisconsin breast cancer data set. The cancer data has two classes, 'Malignant' and 'Benign'. The system treated 'Malignant' as non-self and 'Benign' as self. In order to be sure of providing antigens of novel distributions, self and non-self antigen data was clustered into several groups: the 240 'Malignant' examples were divided into three clusters of 45, 117 and 78 examples, and the 460 'Benign' examples were grouped into three clusters of 42, 355 and 63 examples. The Expectation Maximization (EM) clustering algorithm was applied to cluster antigen data. The EM algorithm is widely-used as the basis for various unsupervised learning algorithms (Mitchell, 1997). 80% of the self and non-self antigen data belonging to each cluster were randomly selected for *N generations*. Therefore, DynamiCS was provided with different antigen data at each generation and the distributions of these data changed at every *N* generations. The costimulation mechanism involving a security officer was implemented by simply increasing the match count only when a detector detects non-self antigens.

3 BACKGROUND: GENE LIBRARY EVOLUTION AND HYPERMUTATION

A problem found in previous experimental results is that the extended DynamiCS required a large number of memory detector co-stimulations in order to obtain satisfactory TP rates (Kim and Bentley 2002b). This problem could originate from the simplification of the developed AIS, which did not adopt all the evolution processes engaged in the human immune system. So far negative selection and clonal selection have already been employed in DynamiCS and their effects were analysed. However, gene library evolution has not yet been adopted in DynamiCS.

The analyses of previous experimental results explained that the extended DynamiCS with high activation threshold of a mature detector (*A)* provided a smaller number of memory detectors and thus it required less involvement from human security officers. However, it missed a larger number of non-self antigens. In addition, they have shown that the generation of more memory detectors by decreasing the *A* can increase TP rates. This was mainly because all the new detectors were generated randomly and thus generated detectors were randomly scattered in the non-self antigen space. In other words, although existing memory detectors detected a sufficient number of non-self antigens to activate, they can be further finely tuned to match more non-self antigens.

If new detectors are generated by taking some feedback from previous detection results into account, then a new detector can be improved to match a larger number of non-self antigens. This idea can be implemented by a model of gene library evolution using hypermutation, as will be described later. Bearing in mind the effect of gene library evolution, this section addresses how the human immune system evolves over generations, and how existing AIS's adopt these mechanisms.

3.1 GENE LIBRARY EVOLUTION BY HUMAN IMMUNE SYSTEMS

The human immune system learns dynamically changing antigens via clonal selection. To be more precise, activating antibodies clone themselves and proliferate across different parts of the body. Cloning antibodies trigger a *somatic hypermutation* process. Somatic hypermuatation mutates a random portion of genes in antibody clones. Mutated offspring of activating antibodies are expected to have wider variations in their antigen-matching genes. Mutants are quickly disseminated across the body and start detecting other types of antibodies. During this process, mutants and existing antibodies compete to detect more antigens and their antigen detection results determine their affinities. The antibodies with higher affinities survive longer and clone themselves more. It is known that clonal selection with hypermutation is essential for the human immune system to permanently learn newly appearing antigens (Paul, 1993; Sompayrac, 1999).

Somatic hypermutation mechanism is distinguished from mutation taking place in a germ line level[1]. While a germ line level of mutation occurs typically at a low rate, mutation applied on activating antibody clones operates at a much higher rate. Another different feature of somatic hypermutation is that it is applied only on a somatic level. It is known that the mutated genes of antibody clones cannot be directly written back to the DNA (or a gene library) of an egg or sperm cell. As a result, the genes of surviving antibody mutants are not passed onto the next generation of the immune system (Paul, 1993; Sompayrac, 1999).

However, it is also known that the learning results via clonal selection with hypermutation during a lifetime indirectly lead the evolution of a gene library in the human immune system over generations. Although the genes of useful antibody mutants are not directly inherited, individuals capable of generating more useful mutants are more likely to survive against various types of pathogens. Thus, the gene libraries of these individuals are passed over generations and offspring having these inherited gene libraries are more likely to have an immune system with a good *capability* of producing useful mutants. This effect was proposed for the first time by Mark Baldwin in 1896 and named as the Baldwin effect (Baldwin, 1896).

While it has been reported that the learning of the human immune system during a lifetime indirectly determines the direction of gene library evolution (Hightower *et al.*, 1996; Perelson *et al.*, 1996), other work by Hightower *et al.* (1995) investigated what determines the direction of gene library evolution (i.e. where the selection pressure of gene library evolution is aimed). This question is about what the evolution strategy of the human immune system is when the goal is that a dynamically changing vast number of antigens should be covered by a much smaller number of antibodies. This work showed that the binary antibodies of AIS evolve toward a balancing point between maximum coverage of the antigen space and the least overlapping coverage of antibody space.

Oprea and Forrest (1998; 1999) advanced further the work by Hightower *et al.* (1995) and studied the diversity required of a gene library in the human immune system, and the role of gene library evolution. This work verified that antibody evolution gets slower and evolves to cover more random antigen niches when the pathogen size (exposed to antibodies) gets smaller. In this case, the immune system does not let the gene library evolve toward existing antigen specific niches. Instead, it evolves toward covering a coarse-grained antigen space. This understanding was drawn from the observation that the survival probability of the organism (the average fitness of immune systems) increased logarithmically with the size of its germ line-encoded antibody repertoire (the number of antibody genes in the library). This result clearly illustrated that the gene library diversity is not maintained for specific recognition of individual pathogens, but rather it evolved to cover a coarse-gain encoding of the regions of the pathogen universe that the species has encountered. A later study by the same author (Oprea, 1999b) investigated the role of hypermutation by investigating its mutating targets. Her experiments showed that hypermutation usually targets to mutate the antigen-binding regions of a gene library and the mutation results often led fine-tuning of antigen-binding parts.

In summary, the gene library evolves by getting indirect feedback from what the human immune system has learned during its lifetime. Germ line diversity that is obtained through gene library evolution is somewhat directed toward covering a coarse-grain antigen space, and learning through hypermuation leads the immune system to fine-tune its detection of the existing antigens.

3.2 GENE LIBRARY EVOLUTION BY ARTIFICIAL IMMUNE SYSTEMS

There are two methods employed by the currently available AIS's in order to evolve their gene libraries. The first approach directs gene library evolution through the Baldwin effect and the second approach allows provision of direct feedback from learning results to a gene library. The first approach initially builds a gene library that is a collection of previously known antibody genes. This initial gene library provides a certain degree of antigen diversity but it obtains a satisfactory level of antigen diversity through gene expression and learning using hypermutation. Although this approach does not directly alter the genes in the gene library, it still allows the gene library to evolve via the Baldwin effect. The second approach often does not distinguish a gene library from an antibody population. It treats a currently existing antibody population as a gene library and thus concentrates on antibody population evolution. As the result, this approach ignores the difference between lifetime learning and evolution over generations, but it emphasises more the study of whether hypermutation accelerates the degree of antibody population evolution, and controls the evolution direction. These two different approaches have been implemented in various ways depending on the adopted AIS model.

One popular group of AIS is the extension of a conventional genetic algorithm. Researchers added several immune features to GA in order to complement some weaknesses found from a conventional GA (Dasgupta *et al.*, 1999a; Hart and Ross, 1999; Gaspar and Collard, 1999; Hajela and Yoo, 1999; Potter and De Jong, 1998; Nikolaev *et al.*, 1999; Michaud *et al.*, 2001). The static clonal selection algorithm introduced in previous work (Kim and Bentley, 2001) belongs to this group. Among these systems, (Hart and Ross, 1999) and (Michaud *et al.*, 2001) used a gene library that is separate from the antibody population. The gene libraries used in these work are collections of some partial solutions and

[1] *Germ line manipulation* requires the altering of the DNA in the reproductive cells which make the fertilized egg, so that the genetic changes will be copied into every cell of the future adult, including his or her reproductive cells.

thus new antibody solutions were generated by concatenating these partial solutions. While Hart and Ross (1999) generated new antibodies using this method exclusively (Michaud *et al.*, 2001) generated only the initial antibody population using a gene library and the antibody population was evolved using a conventional GA. However, neither investigated whether these approaches have additional benefits compared with others that did not differentiate the antibody population from the gene library. These other methods typically generated new antibodies using crossover and mutation operators of GA and antibodies in the population were continuously replaced with evolved new ones (Dasgupta *et al.*, 1999a; Gaspar and Collard, 1999; Hajela and Yoo, 1999; Potter and De Jong, 1998; Nikolaev *et al.*, 1999). From these latter approaches, apart from (Gaspar and Collard, 1999), none of these systems employed hypermutation, which might provide fine-tuned diversity of the antibody population that can cover currently existing antigens. The AIS developed in (Gaspar and Collard, 1999) cloned the best *n* % of antibodies and mutated them with a high rate. From these mutated antibodies, only ones having improved fitness values were entered to the antibody population for selection. They did not study the effect of hypermutation in terms of antibody evolution.

Another popular type of AIS, which use network theory, usually apply a mutation operator to *n* % of best antibodies in an antibody network, and mutated antibodies are tested whether it is added to an existing immune network (Timmis, 2001; Fukuda *et al*., 1998; Watanabe *et al.*, 1998; Ishida, 1996; Lee *et al.*, 1999). From these AIS's, Timmis (2001) and Fukuda *et al.* (1998) did not use a gene library to create initial antibody nodes while others (Fukuda *et al.*, 1998; Watanabe *et al.*, 1998; Lee *et al.*, 1999; Ishida, 1996) initialised antibody nodes with already known local solutions, which can be regarded as a gene library. The systems using a gene library typically developed an artificial immune network in order to find a global solution under a dynamically changing environment by finding an optimal combination of existing local solutions as a global solution. Among these systems, Timmis (2001), Fukuda *et al.* (1998), and Lee *et al.* (1999) applied a high rate of mutation when cloning new antibodies, and only Timmis (2001) investigated the different effects according to different rates of mutation. In this work, he has shown that the network connectivity declined as the mutation rate got higher and thus contributes to increasing the diversity of the antibody network.

Other work by (De Castro and Von Zuben, 2000; De Castro and Von Zuben, 2001) developed an AIS by mimicking exactly the clonal selection process without differentiating the gene library and the antibody population. When this system cloned new antibodies, it applied various mutation rates to each antibody depending on its affinity. It assigned smaller mutation rates when affinity is higher with the intention of increasing the diversity by correcting poorly performing antibodies. However, this work neither investigated the effect of mutation on the antibody population evolution, nor the need to have a separate gene library to accelerate antibody evolution.

4 EXTENDED DYNAMICS: SIMULATING GENE LIBRARY EVOLUTION USING HYPERMUTATION

4.1 ALGORITHM DESCRIPTION

The problem found from previous experiment results was that the extended DynamiCS obtained high TP rates only when it produced a large amount of memory detector co-stimulation. In contrast, for the case having a smaller amount of memory detector co-stimulation, extended DynamiCS struggled to show high TP rates. However, the related work introduced in the previous section suggests that applying hypermutation to immune cells for cloning is a necessary mechanism to fine-tune current immune cells to target non-self antigen binding regions. As a way of resolving the problem of excessive co-stimulation, extended DynamiCS applies this mechanism.

```
If (immature detector population size +
    mature detector population size
     < non-memory detector pop size)
{
     Do
     { if ( number of deleted memory detectors > 0 &&
            mutation rate != 0 )
       { Select a deleted memory detector randomly and
         create its mutant
         Add this mutant to immature detector population.
       } else
         Generate a random detector and
         add it to an immature detector population

     } Until (immature detector population size +
              mature detector population size =
              non-memory detector pop size)
}
```

Figure 1. Modified Pseudo Code for Extended DynamiCS

It can be interpreted that low TP rates obtained by the extended DynamiCS were originated from coarse-grained non-self antigen niche coverage of activating detectors. Thus, if these detectors were more fine-tuned to cover existing non-self antigens, the extended DynamiCS could have higher TP rates without necessarily having a large amount of activating detectors. In order to investigate the effect of hypermutation only, the extended DynamiCS does not create a separate gene library (i.e., a collection of useful detector genes). Instead, it continues to maintain three detector populations: immature, mature and memory detector populations and treats a portion of the memory detector population as a gene library. In order to let memory detectors evolve towards existing non-self antigens without binding self antigens, the extended DynamiCS clones memory detectors by applying a hypermutation operator on deleted memory detectors. These mutants of deleted memory detectors are added to an immature detector population for the negative selection test. Immature detectors in DynamiCS have always been

randomly generated for negative selection. Now extended DynamiCS produces immature detectors by mutants of deleted memory detectors, if there are memory detectors available or by random otherwise. Hence, this further extension of DynamiCS employs a "virtual gene library" dynamically made from mutations of deleted memory detectors. Through the various selection mechanisms and hypermutation operator, the seed immature detectors produced by the virtual gene library evolve over time, just as the immature, mature and memory detectors evolve in their separate populations. This modification is summarised in the pseudo code shown in figure 1.

While the mutation rate used in GAs is very low (around 0.01~0.05%), extended DynamiCS employs much higher rates (0.1% and 0.2%) for hypermutation. This also follows the mutation strategy of the human immune system. The human immune system deliberately uses a higher mutation rate in order to maintain its diversity (Paul, 1993). Similarly, adopting a higher rate of mutation is expected to lead detectors to explore new non-self antigen niches and thus escape from existing self antigen niches. The following sections will study how an unusually larger mutation rate affects the performance of extended DynamiCS.

It also should be noted that hypermutation is applied to deleted memory detectors, not to existing memory detectors. This part is a slight variation of the human immune system. The human immune system clones successful memory detectors and spreads them to other lymph nodes distributed in the body. These new cloned detectors are expected to detect associative non-self antigens that share some non-self antigen patterns detected by previously detectors but do not necessarily have the same non-self antigen patterns with the previous detectors. In other words, cloned detectors are expected to detect new antigens belonging to a new antigen cluster as soon as possible. During this process, the self-tolerance of new mutants are maintained by the helper T-cells. However, extended DynamiCS does not have a separate helper T-detector population to confirm self-tolerance of newly cloned detectors. Therefore, extended DynamiCS uses hypermutation in a way to generate new detectors more tuned to target non-self antigen detection, and at the same time still effectively avoid self antigen detection. Memory detectors are deleted when they match self antigens of the current antigen cluster, but the fact that they managed to become memory detectors at all implies that they hold valid information about non-self antigens in previous clusters. By mutating these and reusing them in the form of a virtual gene library to seed new immature detectors, this evolved information is being retained and fine-tuned by the system.

4.2 EXPERIMENT RESULTS

Two series of experiments were performed in order to investigate the effects of hypermutation on true positive (TP) and false-positive (FP) rates by the extended DynamiCS introduced here. These experiments had the same values of given parameters that were used in the experiments of previous work (Kim and Bentley 2002b), which are summarised in table 1.

Table 1. Parameter values used for DynamiCS experiments

Parameters	Values
Tolerisation Period (T)	30
Life Span of Mature Detectors (L)	10
Activation Threshold of Mature Detectors (A)	{5, 10, 20, 40}
Number of Generations that Antigens are Selected from the Same Cluster (N)	30

The first series of experiments was performed by varying *A* values with mutation rate = 0.1 and the second series was performed with mutation rate = 0.2. Figure 2 and 3 show the average TP and FP rates of each series of experiments after running them five times. The X-axes of these graphs represent the number of generations and the Y-axes indicate detection rates. Each graph has two lines, one displaying a True Positive (TP) rate and another showing a False Positive (FP) rate. The grid lines on the X-axis were placed at every *N* generations for *N* = 30. Each experiment was also run for maximum 2000 generations.

Table 2. Average numbers of surviving, generated and deleted memory detectors during 2000 generations, and average number of memory detector costimulations per generation for the extended DynamiCS with mutation rate = 0.1. The mean values are followed by the variances in parentheses.

Extended DynamiCS with Mutation Rate = 0.1				
	Surviving Memory Detectors	Generated Memory Detectors	Deleted Memory Detectors	Memory Detector Co-stimulation per generation
A = 5	45.5 (21.67)	535.5 (8869.67)	490 (8448.67)	40.48 (14.35)
A=10	37 (4)	376 (1444.67)	339 (1456.67)	31.39 (1.43)
A=20	32.5 (7)	259.5 (176.33)	227 (172)	28.08 (2.99)
A=40	27.5 (24.5)	203.5 (14964.5)	176 (13778)	22.56 (6.66)

Table 3. Average numbers of surviving, generated and deleted memory detectors during 2000 generations, and the average number of memory detector costimulations per generation for the extended DynamiCS with mutation rate = 0.2. The values in parentheses are variances.

Extended DynamiCS with Mutation Rate = 0.2				
	Surviving Memory Detectors	Generated Memory Detectors	Deleted Memory Detectors	Memory Detecor Co-stimulation per generation
A = 5	44.75 (8.25)	264.5 (94.33)	219.75 (88.25)	39.15 (10.52)
A = 10	32.75 (24.92)	193.5 (539)	160.75 (393.58)	27.52 (14.62)
A = 20	29 (8.67)	126.5 (53.67)	97.5 (67)	24.48 (6.94)
A = 40	19.5 (0.33)	98 (1078)	78.5 (1013)	16.75 (1.12)

The effects of hypermutation are clearly revealed when these results are compared to the results obtained in the previous work (Bentley and Kim 2002b). From figure 2 and 3, FP rates are consistently low except one case where A = 5 and mutation rate = 0.2. The differences in TP rates depending on different mutation rates are clearly noticeable when A has a larger value. For instance, when A is 40 without mutants of memory detectors, TP rates ranged between 0.5 and 0.9. On the other hand, when A is 40 with mutation rate = 0.2, TP rates increase so that they range between 0.85 and 0.95 (see figure 3). More importantly, the improvement in TP rates was obtained without increase of FP rates. The scale of TP rate increase is much more noticeable when mutation rate is 0.2 although TP rate increase can be seen when A is 40 with mutation rate = 0.1 (see figure 2). Thus, it is verified that hypermutation affects the result of extended DynamiCS in a positive way: TP rates increase while maintaining low FP rates.

In addition, when A = 40 with mutation rate 0.2 which shows high TP rates and low FP rates, extended DynamiCS still maintained the average number of memory detector co-stimulation per generation as small as seen previously, when mutants of memory detectors were absent (Kim and Bentley 2002b). This result can be found from table 2 and table 3. They show the total number of surviving, generated and deleted memory detectors for total two thousand generations when mutation rate is 0.1 and 0.2 respectively. These numbers are the average numbers of five runs. For both cases, extended DynamiCS had the smallest number of memory detector co-stimulation when A = 40. Furthermore, when the extended DynamiCS had a larger mutation rate, 0.2, it performed less memory detector co-stimulation than when it had a mutation rate 0.1.

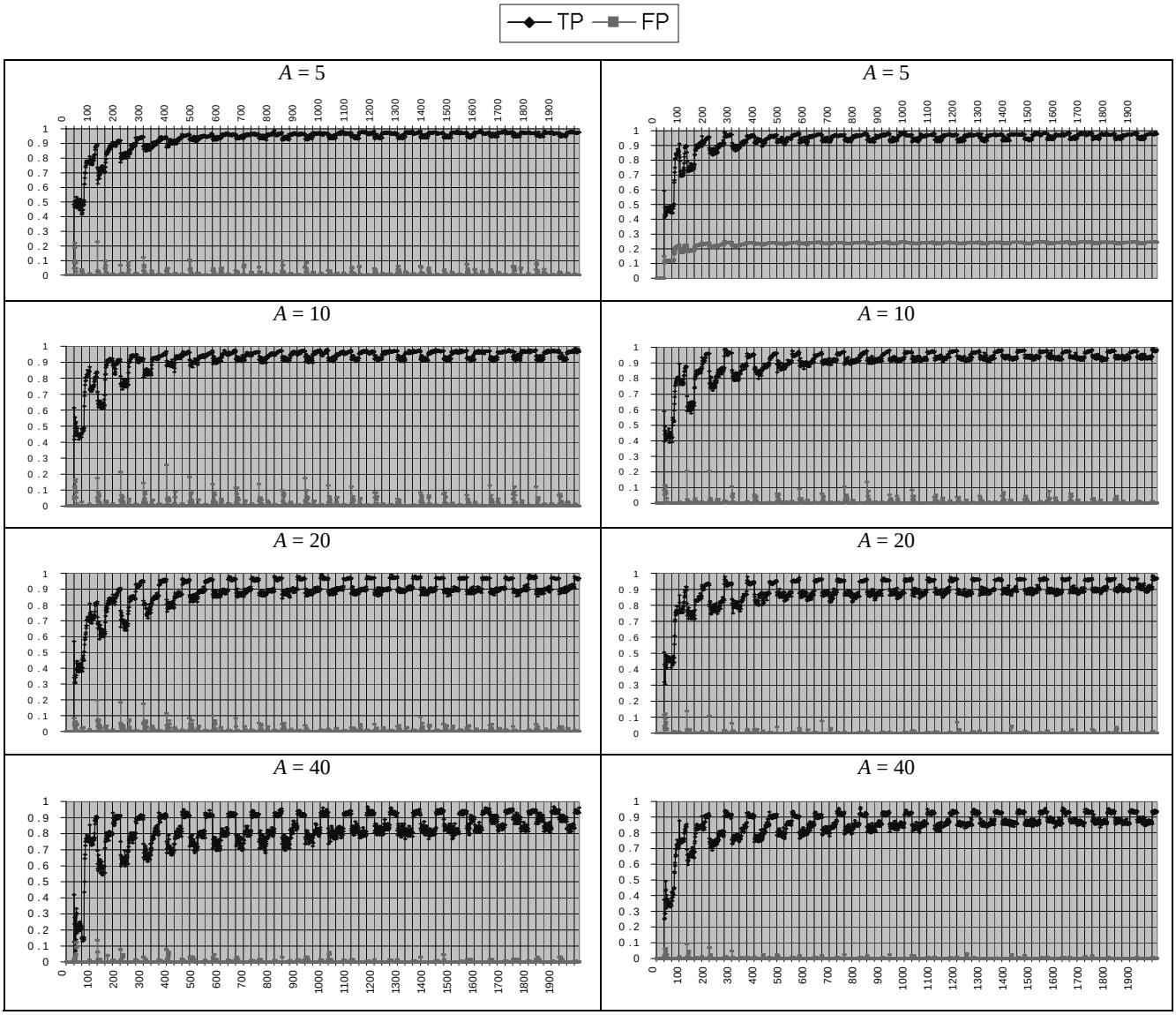


Figure 2. TP and FP rates when A varies and T = 30, L = 10, N =30 with mutation rate = 0.1

Figure 3. TP and FP rates when A varies and T = 30, L = 10, N =30 with mutation rate = 0.2

To summarise, two series of experimental results show that TP rates increased when immature detectors were generated by applying a hypermutation operator to deleted memory detectors. Furthermore, it maintained low FP rates and the small number of memory detector co-stimulation. These positive effects were more clearly found when a larger mutation rate was applied.

5 DISCUSSION OF DYNAMICS

DynamiCS has been introduced to make our AIS fulfil two properties required by an effective intrusion detection system: learn stabilised self behaviours when presented with only a small subset of self antigens at one time and learn sudden changes in converged self behaviours. In order to provide these features to the AIS, DynamiCS employed several novel components such as immature, mature and memory detector populations, tolerisation period, activation threshold, mature detector life-span, mature and memory detector co-stimulation and applying hypermutation to generate immature detectors. All of these novel components were designed by following the mechanisms existing in the human immune system and thus led the AIS to yield desired two properties.

Many of these novel components are based on the different AIS, called LYSIS, proposed by (Hofmeyr, 1999; Hofmeyr and Forrest, 2000). LYSIS is also equipped with three detector populations (immature, mature and memory), tolerisation period, activation threshold, co-stimulation and mature detector life-span. (Hofmeyr, 1999; Hofmeyr and Forrest, 2000) tested LYSIS system against network traffic headers collected for 50 days, consisting of 3900 unique self strings. In order to scale this size of self strings, (Hofmeyr, 1999; Hofmeyr and Forrest, 2000) developed LYSIS in a distributed environment and thus fifty different hosts generated total 5000 immature detectors per day. Similar to DynamiCS, LYSIS also dynamically generated immature detectors and started to monitor new antigens after the first tolerisation period. Although this system was tested against real network headers, the real environment scenario given to these tests was only limited to the first real environment scenario studied in this work: learn stabilised self behaviours with only a small subset of self antigens at one time. Thus, DynamiCS is the only AIS that employed novel components introduced in this work and has been tested on another important IDS real scenario: learn quickly any sudden changes in converged self behaviours. Under this scenario, DynamiCS was capable of detecting non-self antigens in a satisfactory level without losing its self-tolerance and this was achieved by applying hypermutation, which is not adopted by LYSIS.

(Hofmeyr, 1999; Hofmeyr and Forrest, 2000) investigated a way to tune LYSIS behaviours to get desired TP and FP rates. This study was focused on choosing an appropriate tolerisation period, activation threshold and decay rate. It should be noted that the decay rate used in LYSIS was not adopted by DynamiCS. It was regarded that the number of parameters used in DynamiCS already seemed to be large enough to make controlling system behaviour difficult. Although a decay rate was introduced in LYSIS in order to replace detectors in a more dynamic way, DynamiCS managed to provide a similar effect without this parameter by using a gene library evolution model with hypermutation.

6 CONCLUSION

As one way to decrease the poor FP rates caused by memory detectors, DynamiCS was extended by eliminating memory detectors when they showed a poor degree of self-tolerance to new antigens (Kim and Bentley, 2002a). This extended system was tested to determine whether surviving memory detectors no longer caused seriously high FP error rates or not. The test results showed that deletion of memory detectors based on their self-antigen detection dramatically decreased high FP rates. However, this method required a larger amount of co-stimulation in order to gain such benefits. The large amount of co-stimulation can render the system weak for intrusion detection. This disadvantage demanded further extension of DynamiCS.

In order to resolve this problem, this paper explored the use of hypermutation in DynamiCS to produce the effect of gene library evolution. This additional extension was designed to fine-tune generated memory detectors so that the system obtained higher TP rates without increasing the amount of co-stimulation. The gene library evolution was modelled by producing immature detectors via hypermutation on deleted memory detectors. Thus a “virtual gene library”, made from mutations of deleted memory detectors was maintained. The new extension was tested to determine whether it achieved high TP rates without increasing the amount of co-stimulation. The test results confirmed that hypermutation enabled the evolution of the virtual gene library and thus produced immature detectors that were better tuned to cover existing non-self antigens.

References

Kim, J. and Bentley, P. J. (2002a) Towards an Artificial Immune System for Network Intrusion Detection: An Investigation of Dynamic Clonal Selection. *Proceedings of Congress on Evolutionary Computation*, pp.1015-1020, 2002.

Kim, J. and Bentley, P. J. (2002b) Immune Memory in the Dynamic Clonal Selection Algorithm. Submitted to the first International Conference on Artificial Immune Systems (ICARIS).

Hofmeyr, S., (1999) *An Immunological Model of Distributed Detection and Its Application to Computer*

Security, PhD Thesis, Dept of Computer Science, University of New Mexico, 1999.

Hofmeyr, S. A., and Forrest, S., (2000) "Architecture for an Artificial Immune System", *Evolutionary Computation*, Vol. 7, No. 1, Morgan-Kaufmann, San Francisco, CA, pp. 1289-1296, 2000.

Paul, W. E., (1993) "The Immune System: An Introduction", *Fundamental Immunology* 3rd Ed., W. E. Paul (Ed), Raven Press Ltd, 1993.

Sompayrac, L., (1999) *How the Immune System Works*, Blackwell Sicence, 1999.

Baldwin, J. M., (1896) "A New Factor in Evolution", *American Naturalist*, Vol.30, pp.441-451.

Hightower, R., Forrest, S., and Perelson, A. S., (1996) "The Baldwin Effect in the Immune System: Learning by Somatic Hypermutation", in R.K. Belew and M. Mitchell, (eds.), *Adaptive Individuals in Evolving Populations*, Addison-Wesley, Reading, MA, pp. 159-167, 1996.

Perelson, A. S., Hightower, R., and Forrest, S., (1996) "Evolution and Somatic Learning in V-Region Genes", *Research in Immunology*, Vol. 147, pp. 202-208.

Hightower, R., Forrest, S., and Perelson, A. S., (1995) "The Evolution of Emergent Organization in Immune System Gene Libraries", *Proceeding of the Sixth International. Conference. on Genetic Algorithms*, L.J. Eshelman (Ed.), Morgan Kaufmann, San Francisco, CA, pp.344—350, 1995.

Oprea, M., and Forrest, S., (1998) "Simulated Evolution of Antibody Libraries Under Pathogen Selection", *Proceeding of IEEE International Conference on Systems, Man and Cybernetics*, 1998.

Oprea, M. and Forrest, S., (1999) "How the Immune System Generates Diversity: Pathogen Space Coverage with Random and Evolved Antibody Libraries.", *Proceeding of Genetic and Evolutionary Computation Conference (GECCO)*, July,1999.

Dasgupta, D., Cao, Y., and Yang, C., (1999) "An Immunogenetic Approach to Spectra Recognition", *Proceeding of Genetic and Evolutionary Computation Conference (GECCO' 99)*, July 13-17, pp149-155, 1999.

Hart, E. and Ross, P., (1999) "An Immune System Approach to Scheduling in Changing Environments", *Proc. of Genetic and Evolutionary Computation Conference (GECCO'99)*, pp.1559-1566.

Gaspar, A., and Collard, P., (1999) "From Gas to Artificial Immune Systems: Improving Adaptation in Time Dependent Optimisation", *Proceeding of CEC99*, 1999.

Hajela, P., and Yoo, J. S., (1999) "Immune Network Modelling in Design Optimization", in *New Ideas in Optimization*, (Eds.) D. Corne, M. Dorigo, & F. Glover, McGraw Hill, London, pp.203-215, 1999.

Potter, M. A. and De Jong, K.A., (1998) "The Coevolution of Antibodies for Concept Learning", *Proceeding of the fifth Intl. Conference on Parallel Problem Solving From Nature*, pp.530-539, 1998.

Mitchell, T., (1997) *Machine Learning*, McGraw-Hill, 1997.

Nikolaev, N., Iba, H., and Slavov, V., (1999) "Inductive Genetic Programming with Immune Network Dynamics", *Advances in Genetic Programming 3*, MIT Press, Chapter 15, pp.335-376, 1999.

Michaud, S. R., *et al.*, (2001) "Protein Structure Prediction with EA Immunological Computation", *Proceeding of Genetic and Evolutionary Computation Conference (GECCO'2001), July 7-11*, pp.1367-1874, 2001.

Kim, J. and Bentley, P. J. (2001). Towards an Artificial Immune System for Network Intrusion Detection: An Investigation of Clonal Selection with a Negative Selection Operator. In Proc. of *CEC2001, the Congress on Evolutionary Computation*, Seoul, Korea, May 27-30, 2001. pp. 1244-1252.

Timmis, J., (2001) *Artificial Immune Systems: a Novel Data Analysis Technique Inspired by the Immune Network Theory*, PhD Thesis, Dept. of Computer Science, University of Wales, Aberystwyth, 2001.

Fukuda, T., Mori, K., and Tsukiyama, M., (1998) "Parallel Search for Mutil-Modal Function Optimization with Diversity and Learning of Immune Algorithm", *Artificial Immune Systems and Their Applications*, (Ed) Dasgupta, D., Springer-Verlag, Berlin, pp.210 – 220, 1998.

Watanabe, Y., Ishiguro, A., Shirai, Y., and Uchikawa, Y., (1998) "Emergent Construction of Behavior Arbitration Mechanism Based on the Immune System", *Proceeding of ICEC'98*, pp.481-486, 1998.

Ishida, Y., (1996) "An Immune Network Approach to Sensor-Based Diagnosis by Self-Organization", *Complex Systems*, Vol. 10:1, pp. 73-90.

Lee, W., Park, C., and Stolfo, S. J., (1999) "Towards Automatic Intrusion Detection Using NFR", to appear in the *Proceeding of 1st USENIX Workshop on Intrusion Detection and Network Monitoring*, 1999.

De Castro, L. N., and Von Zuben, F. J., (2000) "The Clonal Selection Algorithm with Engineering Applications", *Proceeding of Artificial Immune System Workshop, Genetic and Evolutionary Computation Conference (GECCO' 2000)*, pp36-37.

De Castro, L. N., and Von Zuben, F. J., (2001) "AiNet: an Artificial Immune Network for Data Analysis", (Book chapter in) *Data Mining: A Heuristic Approach*, (Eds) Abbass, H. A., Sarker R. A., Newton, C. S., Idea Group Publishing, 2001.

From Optimization to Learning in Changing Environments: The Pittsburgh Immune Classifier System

Alessio Gaspar and Béat Hirsbrunner
PAI Group, Computer Science Department University of Fribourg (DIUF), Switzerland)
http://www.unifr.ch/diuf/pai/ [alessio.gaspar—beat.hirsbrunner]@unifr.ch

Abstract

A simple computational model of secondary immune response is used to provide a Pittsburgh style classifier system with the ability to improve its reaction to already encountered situations in a dynamical cyclic learning environment. Main results obtained with our core algorithm (YaSais) on Time Dependent Optimization problems are briefly reminded before to introduce the *Pittsburgh Immune Classifier System* (PICS) which is then experimentally evaluated on both a static and dynamical multiplexer problem. Eventually, the Lazy Optimality Effect, keystone of YaSais' efficiency, is re-examinated in PICS. Suggested enhancements are then experimentally evaluated.

1 Introduction

1.1 Motivation, Previous work

While it is commonly admitted that evolutionary algorithms are *adaptive computation* approaches, their convergence limits their adaptiveness. Diversity loss eventually disables the crossover effects and leaves mutations as the only exploration drive. Consequently, Evolutionary Time Dependent Optimization (ETDO) is often used as a benchmark for adaptiveness [14, 13, 16]. This framework led us to suggest three necessary (but not sufficient) properties to characterize the adaptiveness in changing environments [4]:

Reactiveness: ability to recover from transitions and find the new optimum.

Robustness: ability to limit the loss in the best fitness value featured by the population when the transition occurs.

Immunization: ability to improve the robustness when undergoing a transition to an already encountered optima.

So far, we focussed on evaluating a Simple Artificial Immune System for Evolutionary TDO. The results were interesting and encouraged us to investigate how such a basic algorithm can perform in learning problems by combining it with classifier systems with the objective to provide the latter with an immunization capability.

1.2 Objective Statements

We are switching from Time Dependent Optimization (TDO) to Time Dependent Learning (TDL) problems and evaluating, along the way, a simple, general purpose, immune algorithm. Next section introduces the YaSais algorithm [6, 4], sums up previous results and details the *Lazy Optimality Effect*, keystone of its efficiency in TDO. Section 3 introduces the Pittsburgh Immune Classifier System (PICS) as a combination of YaSais with a Pittsburgh Classifier System (PCS). Preliminary results on a static and then dynamic multiplexer problem (MUX) are compared to those obtained by YaSais's on TDO. Section 4 further details how specific evolutionary effects present in YaSais introduced unexpected results in PICS. Suggestions as to how to improve PICS are then evaluated. Section 5 concludes by discussing analogies with latent learning classifier systems.

2 YaSais: the core immune algorithm

2.1 Generalities

This section describes YaSais (Yet Another Simple Artificial Immune System), an improved version of Sais algorithm [5], and reviews the most important results (immunization, LOE) needed to ground our later discussion on PICS.

To quickly locate YaSais among Evolutionary Algorithms, let's describe it as a Genetic Algorithm which K-Tournament selection has been modified in order to select only some individuals to be cloned and then used to perform exploration (crossover and high mutation rate are applied), and which favors good parents vs. mediocre offsprings during recruitment. The main differences are (1) explicit clustering of the population into *gatherings*, (2) selection of individuals to be cloned while others are kept unchanged (clonal selection) and (3) use of intensive exploration techniques (somatic hypermutation) on clones.

To be more accurate, YaSais's key idea is to divide the population into $\mathcal{G}$ equi-sized *gatherings* of B-Cells[1]. The selection mechanism decides which B-Cell(s) per gathering will be *activated* and serve as a basis for further exploration. This approach is loosely inspired by Jernes' Idiotypic Networks theory [9, 10] on immune system's memory.

Simply stated, B-Cells[2] can be activated by antigens (when directly useful against one of them) or by other B-Cells (anytime). Therefore, if B-Cell A activates B which activates C which in turn activates A, we have a self reinforcing dynamics. Each B-Cell's activation, and therefore reproduction, is ensured in an endogenic way and memorizing boils down to integrating B-Cells into such idiotypic cycles.

Evolutionary Algorithms inspired by this theory bend the evolutionary dynamics so that it is not only convergent but also maintains stable subpopulation with respect to other fitness criteria than optimality in the current environment (eg. previous optimality in TDO).

[1] aka chromosomes in other evolutionary algorithms.

[2] We do not differentiate B-Cells and antibody herein.

2.2 YaSais Algorithm

- **0. Initialization: create** $P(0)$
  - Let $P(0)$ be a population of $|P|$ random B-Cells each λ bits long.
  - Arbitrary, $P(0)$ is divided in $\mathcal{G}$ groups of B-Cells (Gatherings).
  - Generation number t is set to 0.
- **1. Evaluation**
  - For each B-Cell in $P(t)$, compute its fitness. For the Pattern Tracking, it is the complement of its Hamming distance to the current arbitrary chosen optimum.
  - For each Gathering in $P(t)$, mark the best fitted B-Cell. There will be $\mathcal{G}$ B-Cells marked in $P(t)$.
- **2. Clonal Selection:** $P(t) \rightarrow P_{ex}$
  - Create an empty population P_{ex} of size $|P_{ex}| = \mathcal{G} * CF$, where CF is a parameter of the system (the Cloning Factor).
  - Fill it with G B-Cells by K-Tournaments among the ones marked in P_{ex}.
  - Copy each B-Cell added to P_{eX} CF times (cloning).
  - For each clone in P_{ex}, apply high rate random mutations (hypermutating).
- **3. Recruitment:** $P(t) + P_{ex} \rightarrow P(t+1)$
  - For each of the $\mathcal{G}$ marked B-Cells in $P(t)$, select a challenger with a K-Tournament ($K = 3$) in P_{ex} and replace the current B-Cell only if less fitted.
  - Let $P(t+1) = P(t)$
  - Branch to bf 1. (fixed iterations)

2.3 YaSais Algorithm step by step

Evaluation Phase

In a Pattern Tracking problem [13], the optimum is arbitrarily chosen as a point of the search space every g generations. The fitness of each B-Cell is therefore measured as its Hamming distance to the current optimum (thus simulating immune-like matching to a given antigen): $\forall B_i \in$

$P(t), Fitness(B_i) = \lambda - \delta_h(B_i, O^t)$ where B_i is the i^{th} B-Cell of $P(t)$, O^t the optimum at time t, λ the length of its binary code and δ_h the Hamming distance. Every $\delta_t = 50$ generations (*transition period*), a new optimum is randomly chosen at a Hamming distance δ_d from the previous one (*transition distance*). This evaluation also enables us to mark the n best fitted B-Cells in $P(t)$ (n being an heuristic value).

Pattern Tracking can be seen as the dynamical counterpart of the 0-max problem which has been widely used to understand genetic algorithms. The reasons for choosing this benchmark are twofold. At first, it is simple from a static point of view which helps in keeping experiments focussed on the dynamical difficulty and avoid biases induced by static aspects. Secondly, its parameters can be set to feature a specific dynamical difficulty [3]. This helps evaluating YaSais on well understood and controlled difficulty levels.

Clonal Selection Phase
This phase mimics the core of the immune system's evolutionary dynamics [8]: cloning the B-Cells matching antigens. We pick up the $\mathcal{G}$ best B-Cells from $P(t)$ and clone them CF (*Clonal Factor* parameter) times each to obtain the temporary population P_{ex}. Then, we simulate *Somatic Hypermutation*[3] by randomly mutating each member of P_{ex} and preserving only the mutants improving fitness.

Recruitment Phase
Eventually, we reintroduce worthy B-Cells from P_{ex} into $P(t)$ in order to build $P(t+1)$. The B-Cells that have not been involved in the building of P_{ex} remain unchanged so that they can implement an implicit memory of past optima. The marked B-Cells are compared to the winner of a K-Tournament ($K = 4$) in P_{ex} and only replaced if being less fitted.

This approach both guarantees stability of the densities of previous optima which fitness is of no interest anymore, and an elitist dynamics which forbids the best fitness featured at next generation to be lower than the current one.

[3]Natural Somatic Hypermutation mutates the DNA of B-Cells resulting from clonal selection [8].

2.4 Previous Experimental Results

We briefly sum up previous experimental results obtained with YaSais on a Cyclic Pattern Tracking (PT) problem [13, 16] with a focus on its immunization capability only.

In a Cyclic Pattern Tracking (CPT), a list of n successive optima is defined (δ_t fixed for all). An epoch is a duration of $n * \delta_t$ generations during which all optima are presented. Epochs follow each other and thus enable us to evaluate YaSais' reaction to already encountered transitions.

YaSais features a tradeoff between reactivity and robustness [6]. Most evolutionary TDO solutions trade a good robustness for a high fitness level or *vice et versa*. By comparing YaSais to robust [2, 15] and reactive [17, 1, 7] algorithms, we underlined that YaSais is equivalent in terms of efficiency to method up to 4 times more computationally expensive which dominate other evolutionary algorithms that were compared in [4].

On the other hand, YaSais featured an immunization capability which is illustrated by Figure 1. This experiment was averaged over 50 runs for a length of 1000 generations (4 epochs). YaSais ($CF = 4$,$\mathcal{G} = 8$, $K = 4$, $|P| = 40$, $\lambda = 40$, $X_c = 0.7$, $\mu = 0.01$) was applied to a Cyclic Pattern Tracking problem with 5 optima ($\delta_t = 50$ and $\delta_d = 5$ then increased by 5 at each transition).

The upper part of the Figure plots the best fitness per generation. The fitness loss at transitions is reduced over consecutive epochs which is the sign of an ongoing immunization. In the lower part of the Figure, the densities of the 5 successive optima used in this environment are plotted. This complements the previous information by showing the number of copies of each optimum grow during the period at which is it the current optimum. Moreover, these density curves also show that once non longer the current optimum, each of these individuals is kept in the population.

2.5 The Lazy Optimality Effect (LOE)

So YaSais features an immunization capability but all optima are not memorized durably in

the population and fitness keeps dropping slightly when they are encountered again. Why ?

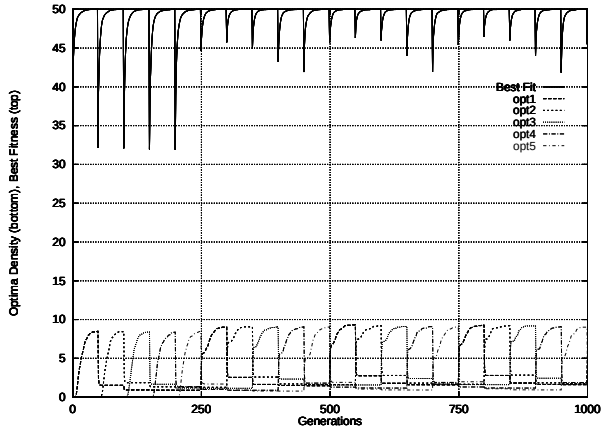


Figure 1: YaSais / Cyclic Pattern Tracking
Top: best fitness, Bottom: 5 optima's densities

At each generation, $|P| - \mathcal{G}$ B-Cells are kept unchanged and $\mathcal{G}$ are chosen to initiate an intensive search. This mechanism is responsible for loosing previous optima. By taking a closer look to transitions in a single-run experiments (CPT, $\mathcal{G} = 8$ gathering, 5 B-Cells each) we observed the following pattern: the density of the current optimum decreases suddenly (eg. 6 B-Cells) and keeps doing so (less significantly though) during consecutive transitions. On the other hand, density of the next optimum increases from 2 to 8 B-Cells (same example transition).

Why are the B-Cells encoding the current optimum more often selected ? Quite simply, they are (on average) the closest to the new optimum in the population. Remember that in our CPT problem the n optima are determined as follow: B-Cells are divided in λ/n bit-long blocks, 1st optimum is all '0' except for '1' filling the 1st block, 2nd optimum has 2nd block set to '1' and so on. Consequently, the Hamming distance between consecutive optima is constant and equals to 20 bits ($n = 5$ and $\lambda = 50$).

We know that YaSais will mark the closest B-Cells to the new optimum. The probability of a random string to match the new optimum is 0.5^{λ} and the probability for it to be located at a Hamming distance less or equal to 20 is $P = \sum_{d=0}^{20} 0.5^{\lambda}.C_{\lambda}^{d}$. That is, the probability for a non-previously optimal B-Cell (assumed to be random) to be located closer to the new optimum than any previous optimum is $P = 0.1$ in our case.

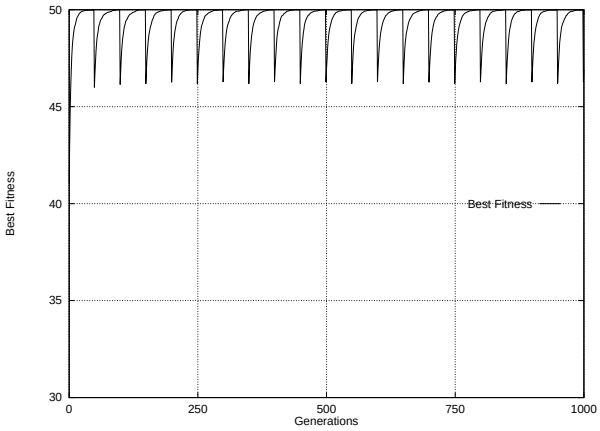
We checked this on a transition in the previous experiment. At generation 1000, YaSais lost 6 B-Cells encoding previous optimum and gained 6 B-Cells encoding new optimum (from 2 to 8). We also counted 17 instanced of current or previous optima. Among the 23 remaining B-Cells, 3 only ($P = 0.1$) have a chance to be selected instead of previous optima. Knowing that $\mathcal{G} = 8$ are going to be picked up, even if all 3 are retained, 5 out of 6 B-Cells encoding previous optimum should be used for exploration (6 B-Cells were used).

Therefore, if two consecutive optima are close enough, the system forgets about the previous one but still features an overall good performance. Why ? When the distance is short, previous optima are lost but with limited consequences since finding the new one is simple enough. Otherwize, if the transition gets more difficult, the immunization plays its role. We termed this the Lazy Optimality Effect (LOE), since immunization is only used when nothing simpler works.

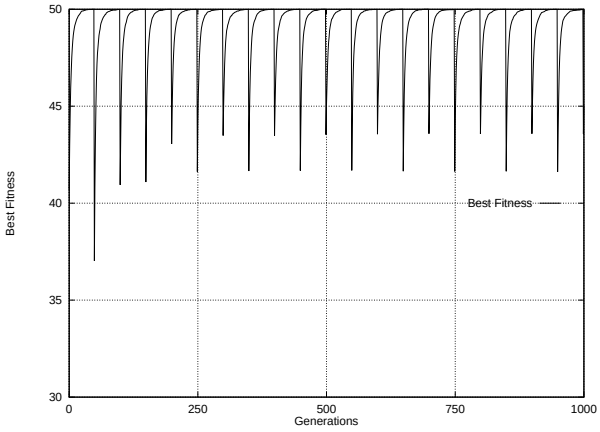
We replicated previous experiment with only two optima and varied their relative distance to check the influence of this factor. Figure 2 confirms that for a high distance (12 and above) the immunization is perfect. On the other hand, results are quite good for a very low distance as well (2) and less good between those two extrema. This is no surprise since a small δ_d minimizes the fitness loss (cf supra) but it is important to understand that this is achieved without immunization. Examining the density curves of each optima confirms that with small δ_d, optima are lost regardless of the misleadingly appealing fitness curve.

2.6 Conclusion: YaSais / TDO

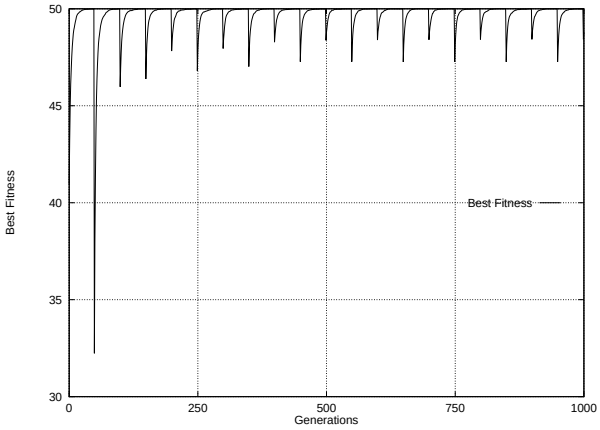
The experimental results on Pattern Tracking revealed that YaSais is an efficient dynamical optimization Tool. A restriction should be kept in mind as we only considered so far non epistasic problems for which we suspect the somatic hy-



Distance 2



Distance 8



Distance 12

Figure 2: YaSais / Cyclic Pattern Tracking Distance Between Optima vs. Immunization

permutation to play a central role in improving the system's reactiveness (cf next section).

YaSais also features an improved robustness to environmental changes; Whenever the transitions are easy (low δ_d), the natural diversity kept in the population is enough to ensure a good level of fitness to be kept during the transitions. On the other hand, when confronted to difficult transitions, YaSais takes advantage of any relevant information in the population such as previous optima. Therefore, an implicit tradeoff is realized between the use of the random diversity and the "oriented one" induced by the immunization process. The rule seems to be "if it is hard to find, remember it, otherwise, just drop it". Even if not reaching a perfect immunization ability as we initially expected, we must admit that, although it is more "lazy", YaSais uses at best its capabilities.

3 PICS Time Dependent Learning

3.1 PICS algorithm

Classifier Systems (CS) have been investigated in two main flavors. Michigan style CS evolve a population of rules which constitute altogether the CS which policy is evaluated in a given environment. If a reward is earned, Reinforcement Learning techniques are used to perform the necessary Credit Assignment among the rules that contributed to the successful behavior.

On the other hand, the Pittsburgh approach is about evolving a population where each individual encodes the ruleset of an independent CS. Fitness is computed by decoding a given individual into a CS and evaluating its interaction with the environment (eg. average reward over a given time). This approach only relies on evolution to find efficient classifiers and is therefore a natural candidate for designing an hybrid algorithm embedding the key features of YaSais.

Therefore, we evolved individuals encoding full CS with YaSais instead of a conventional Evolutionary Algorithm. Our objective is to provide a *cognitive immunity* by preserving previously useful policies in the population. This section details experiments on both static and dynamic multiplexer problems and discusses LOE in a learning context.

3.2 Preliminary Experiments: S-7-MUX

Let us consider a 7 bits instance of the Static Multiplexer Problem (S-7-MUX): we have 6 bits long inputs and 1 bit output. The input is separated in 2 *address* bits and 4 *data bits*. For any input, the correct output is the input data bit located at an index given by the decimal value of the 2 input address bits. For instance, input [10 0010] corresponds to output [1]. Consequently, the followings are the minimal and most generic [11] set of rules solving the 7 bits multiplexer problem:

[00	0###]	→	[0]	[10	##0#]	→	[0]
[00	1###]	→	[1]	[10	##1#]	→	[1]
[01	#0##]	→	[0]	[11	###0]	→	[0]
[01	#1##]	→	[1]	[11	###1]	→	[1]

Ideally, classifier systems should converge toward this rule set. Basic approaches do not most of the time but recent advances help in ensuring the generality of the solutions [18].

We started off by applying PICS to S-7-MUX with the following experimental conditions:
1300 generations, results averaged over 20 runs
B-Cells: $\lambda = 140$ bits encoding 20 rules
Rules: $[2+4] : [1]$ (input = 2 bits address + 4 bits data, output = 1 bit
Population: $|P| = 100$
Evaluations: over 30 input samples (among 64 possible) randomized at each fitness function call
Selection: $K = 2$ (select) $K = 3$ (recruit)
PICS specifics: $\mathcal{G} = 20$, $CF = 3$
Operators: $X_c = 0.8$ (uniform) and $\mu = 0.01$

Figure 3 plots the fitness of the best individual of each generation (upper curve). Knowing that the best reward in this one-step environment is 1000, we can deduce that the approach is performing decently, featuring an asymptotic convergence which is pretty common in evolutionary computation. During single runs, we picked up the best individual at generation 1300 and fed the classifier system it encodes with all 64 possible input. The result of this evaluation of its "coverage" of all perceptions revealed that highly fitted individuals's coverage copuld be as low as 47%.

To be able to measure this phenomenon reliably

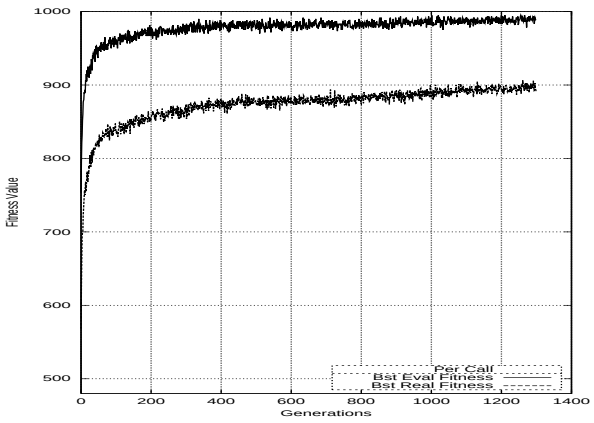


Figure 3: PICS / S-7-MUX
Top: Evaluated Fitness, Bottom: Real Fitness

and understand it better we decided to measure another statistics during the experiment. The lower curve represents the fitness of the best B-Cell of each generation once computed over 300 samples. This value is more representative of the true value of each individual and, as can be seen, is lower than the one featured by the "quick" 30 samples evaluation scheme driving evolution.

Let us keep this issue in mind and move on to the other experiments. Section 4 will revisit these observations, suggest and evaluate a solution.

3.3 D-7-MUX Experiment

This section completes the previous experiment by evaluating PICS immunization capability in a dynamical environment. The dynamical 7 bits Multiplexer problem (D-7-MUX) is similar to its static counterpart. We decided to have a transition period $\delta_t = 2000$ to allow full convergence. Four different environments are going to be presented during one epoch (8000 generations) and then repeated over and over for 4 epochs (32000 generations). The first environment is S-7-MUX. Then, we generated 3 other environments from it by adding a *shift* value σ when decoding the address bits. During first period, $\sigma = 0$ then $\sigma = 1$ and so on up to $\sigma = 3$ after which $\sigma = 0$ again as we start a new epoch. Consequently, input address bits 00 will correspond to the 1st input data bit during 1st period, then to the 2nd during 2nd

period and so on. Let's see how input $[00 \quad 01\dot{0}\dot{1}]$ is multiplexed in the 4 environments[4]:

$$\begin{array}{lllll} \sigma = 0 & [00 & 01\dot{0}\dot{1}] & \rightarrow & [0] \\ \sigma = 1 & [00 & 01\dot{0}\dot{1}] & \rightarrow & [1] \\ \sigma = 2 & [00 & 01\dot{0}\dot{1}] & \rightarrow & [\dot{0}] \\ \sigma = 3 & [00 & 01\dot{0}\dot{1}] & \rightarrow & [\dot{1}] \end{array}$$

Figure 4 also plot the best fitness per generation as evaluated by PICS (upper curve) and accurately evaluated over 300 samples (lower curve). The vertical dotted lines represent transitions from one epoch to another. Other parameters were kept identical to previous experiment. The following observations can be made:

Immunization:
PICS is indeed able to get immunized to previously encountered optima. Both fitness curves progressively reduce their drop off at transitions to new optima over epochs. PICS' core algorithm therefore turned out to be able to feature an identical immunization ability for both TDO and TDL problems which is the first point we wanted to make sure of in this paper.

Resuming Learning:
Both best fitness curves, but especially the lower one, increase from epoch to epoch. After reaching a certain fitness level while solving the first environment ($\sigma = 0$), PICS deal with 3 other environments. When it is again dealing with the first one, its immunization, besides increasing robustness, also enables it to use the $\delta_t = 2000$ generations of the period to improve its fitness level in this environment. It seems to do so from epoch to epoch, "resuming" its learning of each successive optima each time and giving an overall asymptotic trend of improvement.

Fitnesses Differences:
It can also be noticed that the difference between both fitness curves tends to reduce asymptotically over epochs. It can be said that despite the problem underlined in the previous section, PICS manages to overcome it over time as it accumulates information about its environment over epochs instead of converging and discarding any information while re-converging toward another optimum.

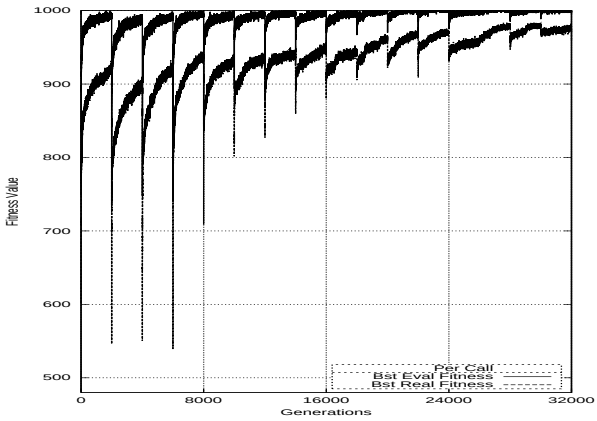


Figure 4: PICS / D-7-MUX
Top: Evaluated Fitness, Bottom: Real Fitness

4 Corrupted Lazy Optimality Effect

4.1 From LOE to CLOE

Our hypothesis is that the LOE is responsible for the observations in Figure 3.

The upper part of Figure 5 illustrates how B-Cells should be specialized. Let us consider one gathering in $P(t)$. Once the fitness function is computed for all its B-Cells, one is activated (selected to be copied into P_{ex}). This B-Cell will be replaced by a better fitted offspring resulting from the exploration performed in P_{ex}. This can be seen as the gathering getting its best fitted B-Cell furthermore specialized to fit the problem at hand. When environment changes, another individual will be specialized to meet its requirements or a previously activated one re-used thus loosing part of its previous specialization (LOE).

The lower part of Figure 5 illustrates what happens in practice when changing the evaluation set every generation. As can be seen, this boils down to changing the fitness function and PICS reacts by specializing another B-Cell (LOE). Conceptually, this is right insofar that, from the evolutionary algorithm standpoint, *Time Dependent* and *Stochastically Evaluated* fitness landscapes are the same: fitness values are altered over time. What

[4]Doted notation $\dot{0}$ and $\dot{1}$ is only used to make the example unambiguous, only a binary alphabet is used.

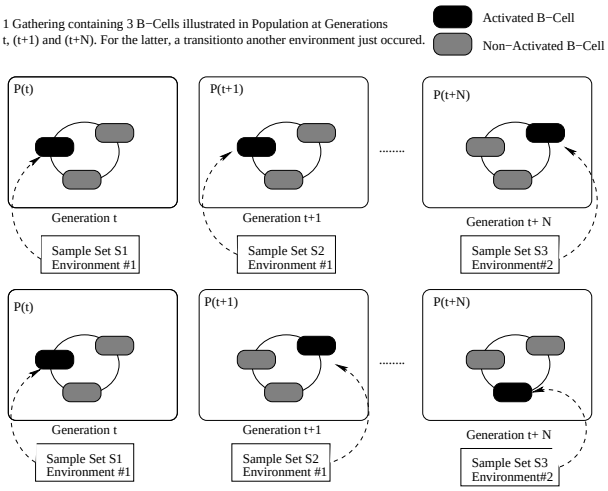


Figure 5: CLOE: specialization of B-Cells
Top: expected activation, Bottom: observed one

causes such a change does not make much qualitative difference even though it may influence difficulty of transitions [3]. Nevertheless, we would like PICS to differentiate between changes in the environment, which call for specialization, and bias due to the stochastic nature of evaluation.

It is worth noticing that in practice the situation is even worse since the evaluation sample is randomized at every fitness function call thus increasing the bias in comparison to the above example. Next section evaluates a way to compensate this Corrupted Lazy Optimization Effect (CLOE).

4.2 Getting to know CLOE better

As previously stated, if we take the best B-Cell produced by a run and evaluate it on all possible 64 inputs, its efficiency is way inferior to what its fitness value promised. This can be seen by taking the best B-Cell of each generation and evaluating its fitness over 300 samples instead of 30.

This suggests that a whole gathering may be able to react correctly to all possible inputs but a single B-Cell is not. While we expected B-Cells of a gathering to specialize into successively optimal policies, it seems they specialized in solving subsets of all possible input samples.

How can we help PICS to specialize only during transitions ? Our working hypothesis is that changing the evaluation set at each generation (as illustrated in Fig. 5) or at each fitness call (as done by PICS) makes a difference. We checked it by changing the stochastic evaluation policy accordingly and decided to randomize the evaluation samples set at each generation and use it for evaluating the whole population.

The top plotting in Figure 6 is similar to Fig. 3. Experimental conditions were identical (S-7-MUX) except concerning evaluation policy. The following observations can be made:

Convergence Time:
It has been shortened from 1400 to 400 generations thus providing the algorithm with a fastest way to handle static learning problems.

Fitnesses Differences:
The sampled fitness values converge sooner toward the ones obtained with a thorough evaluation. This should lead to a better accuracy in efficiency of evolved policies.

Our second hypothesis is that the more two consecutive evaluation sets differ, the more likely it is for another B-Cell to be activated (cf. LOE). Therefore, we introduced the *overlap* parameter: the number of samples kept unchanged from one generation to the next in the evaluation set.

The first plotting in Figure 6 had a null overlap (new evaluation sets at each generation), the second has a maximal value (29 samples are kept unchanged over 30). Results indicate that increasing this parameter degrades efficiency and further separate the real fitness value from the one computed by PICS internally. This clearly invalidates our hypothesis and leads us to conclude that the best evaluation policy is to use the same evaluation set for the whole population and change it completely at each generation to maximize the diversity of samples the system learns from.

5 Conclusion

5.1 Discussion

PICS shows that YaSais core principles can be used to get immunized to successively optimal

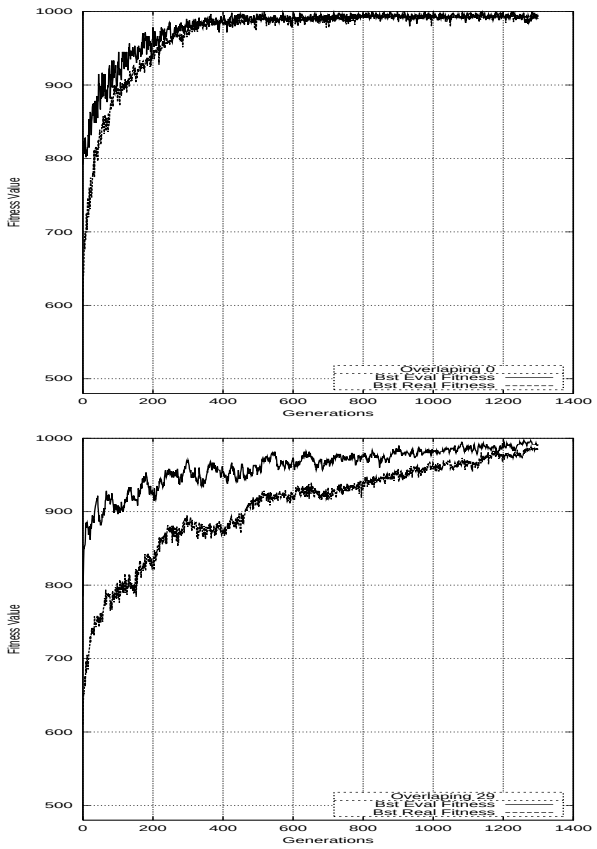


Figure 6: PICS / S-7-MUX
New Evaluation Scheme (overlap parameter)

policies in a TDL problem. The learning problem we investigated can also be seen as a highly epistasic, stochastically evaluated optimization problem and thus as a validation of our immune core algorithm on a more difficult problem than the Pattern Tracking one that we used so far.

An interesting analogy can also be drawn with the latest advances from the Classifier System community concerning *latent learning* approaches. These systems do not only seek for an optimal policy in a given environment but also build progressively a model of the environment which is improved by every trial no matter how wrong or right it is [12]. PICS also models successively optimal policies which, combined altogether, describe the whole environment dynamics. This information could be used by engineers to improve evolved classifier systems or simply understand how the system came to such a solution.

5.2 Synthesis

This paper presented an hybrid algorithm combining an immune algorithm (YaSais) with a Classifier System. The so-called Pittsburgh Immune Classifier System (PICS) has been evaluated in both a static and dynamic Time Dependent Learning (TDL) environment based on the 7 bits multiplexer problem. Preliminary experimental results revealed that PICS features a secondary immune response in its way to discover and memorize optimal policies for various environments. A particular evolutionary effect has been given more attention, explaining efficiency and suggesting a new improvement which was detailed and evaluated on a static environment.

5.3 Perspectives

Current work focuses on CLOE as well as determining the influence of main parameters (CF, $\mathcal{G}$).

A reviewer's suggestion also caught our interest: *Is YaSais suitable for multimodal optimization despite the fact that explicit clustering is intuitively not interesting if no information on the number of peaks is available ?* Each gathering should contain elements from all niches and thus the gathering's size (and number) is critical to ensure they can hold all optima in the multimodal or dynamic environment. This makes setting the $\mathcal{G}$ parameter highly problem-dependent. This argument relies on the assumption that each gathering converges to the same set of optima. If YaSais can evolves differently composed gatherings, their number and sizes become less critial in providing instances of all optima in the population. This remains to be established.

Acknowledgment: *Swiss National Science Foundation grant #20-65301. We are grateful to the reviewers for the above discussion.*

References

[1] H.C. Cobb and J.J. Grefenstette. Genetic algorithms for tracking changing environments. In *Icga-5*, 1993.

[2] J.C. Culberson. Genetic invariance: A new paradigm for genetic algorithm. Technical Report 92-02, University of Alabama, 1992.

[3] A. Gaspar. Etude de l'adaptativite de systemes evolutionnaire en environnement a fitness dynamique. In *Ph.D. Dissertation, University of Nice Sophia Antipolis, July 2000*, 2000.

[4] A. Gaspar. Secondary immune response for time dependent optimization. Technical report, PAI group, DIUF, University of Fribourg (Switzerland), July 2002. 23p.

[5] A. Gaspar and P. Collard. From gas to artificial immune systems: Improving adaptation in tdo. In *CEC-1999: IEEE Congress on Evolutionary Computation*. IEEE society press, 1999.

[6] A. Gaspar and P. Collard. Two models of immunization for time dependent optimization. In *SMC-2000: IEEE International Conference on Systems, Man and Cybernetics. Special Track on Artificial Immune Systems*. IEEE society press, 2000.

[7] J.J. Grefenstette. Genetic algorithms for changing environments. In R. Manner and B. Manderick, editors, *Parallel Problem Solving from Nature 2*, pages 465–501. Elsevier Science Publishers B.V., 1992.

[8] Ron Hightower, Stephanie Forrest, and Alan S. Perelson. The Baldwin effect in the immune system: Learning by somatic hypermutation. In Richard K. Belew and Melanie Mitchell, editors, *Adaptive Individuals in Evolving Populations: Models and Algorithms*, pages 159–167. Addison Wesley, Reading, MA, 1996.

[9] N.K. Jerne. Towards a network theory of the immune system. *Annals of Immunology*, 125(C):373–389, 1974.

[10] N.K. Jerne. Idiotypic networks and other preconceived ideas. *Immunological Reviews*, (79):5–24, 1984.

[11] Tim Kovacs. XCS Classifier System Reliably Evolves Accurate, Complete, and Minimal Representations for Boolean Functions. In Roy, Chawdhry, and Pant, editors, *Soft Computing in Engineering Design and Manufacturing*, pages 59–68. Springer-Verlag, London, 1997.

[12] W. Stolzmann P. Gerard and O. Sigaud. Yacs : a new learning classifier system using anticipation. *Journal of Soft Computing : Special Issue on Learning Classifier Systems.*

[13] K. Pettit and E. Swigger. An analysis of genetic based pattern tracking and cognitive based component tracking models of adaptation. In *Proceedings of National Conference on AI (AAAI-83)*, pages 327–332. Morgan Kaufmann, 1983.

[14] R. Salomon and P. Eggenberger. Adaptation on the evolutionary time scale: A working hypothesis and basic experiments. In *Evolution Artificielle*, pages 297–308, 1998.

[15] R.E. Smith, S. Forrest, and A.S. Perelson. Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation*, 1(2):127–149, 1993.

[16] F. Vavak and T.C. Fogarty. Comparison of steady state and generational genetic algorithms for use in nonstationary environments. In *IEEE International Conference on Evolutionary Computation (ICEC)*, pages 192–195, 1996.

[17] F. Vavak, T.C. Fogarty, and K. Jukes. A genetic algorithm with variable range of local search for tracking changing environments. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature – PPSN IV*, pages 376–385, Berlin, 1996. Springer.

[18] Stewart W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.

Neuro-Immune and Self-Organizing Map Approaches to Anomaly Detection: A Comparison

Fabio González[†] **and Dipankar Dasgupta**
Division of Computer Science
The University of Memphis
[†]and Universidad Nacional de Colombia
{fgonzalz, ddasgupt}@memphis.edu

Abstract

The purpose of this work is to investigate a hybrid approach (neuro-immune technique) for anomaly detection on time series data. In many anomaly detection applications, only positive (normal) samples are available for training purpose. However, conventional classification algorithms need both positive and negative samples. The proposed approach uses normal samples to generate abnormal samples that are subsequently used as training data for a neural network. The approach is compared against an anomaly detection technique that uses self-organizing maps to cluster the normal data sets (samples).

1 Introduction

The anomaly detection problem can be stated as a two-class classification problem: given an element of the space, classify it as *normal* or *abnormal*. Different terminologies are used in different applications, such as "novelty [3] or surprise [13] detection", "fault detection" [20], and "outlier detection" . Accordingly, many approaches have been proposed which include statistical [4], machine learning [15], data mining [16] and immunological inspired techniques [2, 8, 11].

In many anomaly detection applications, however, negative (abnormal) samples are not available at the training stage. For instance, in a computer security application, it is difficult, if not impossible, to have information about all possible attacks. In the machine learning approaches, the lack of samples from the abnormal class causes difficulty in the application of supervised techniques (e.g. classification). Therefore, the obvious machine learning solution is to use an unsupervised algorithm (e.g. clustering).

In our previous work [9], we presented an approach inspired by the immune system that allows the application of conventional classification algorithms to perform anomaly detection tasks. This approach uses a negative selection algorithm (NSA) [6] coupled with a classification algorithm to produce an anomaly detection function. The paper [9] examines the possibility of combine NSA with a neural network classifier in order to detect anomalies in a time series. The purpose of the present work is to perform further experimentation and compare the results to those produced by an unsupervised technique that clusters the normal samples.

The clustering technique used for this purpose is self-organizing maps (SOM) [14]. It is applied to the normal samples to produce clusters that constitute a compact description of the normal space. This compact representation is subsequently used to classify new samples as normal or abnormal [7, 17, 12].

2 Neuro-Immune Technique for Anomaly Detection

The NSA was initially proposed by Forrest and her group [6] based on the principles of self/non-self discrimination in the immune system. It uses as input, a set of strings that represents the normal data (self set) in order to generate detectors in the non-self space. The negative detectors are chosen by matching them to the self strings: if a detector matches it is discarded, otherwise, it is kept. Some efficient implementations of the algorithm (for binary strings) that run in linear time with the size of self have been proposed [5, 6, 11]. However, the time complexity of these algorithms is exponential on the size of the matching window (the number of bits to use in the comparison of two binary

strings).

We proposed [9] a new version of the NSA that represents the self/non-self space as n-dimensional real vectors. One of the advantages of this approach is that it is easier to extract meaningful knowledge from the generated detectors as the representation is closer to that of the problem space. The detectors generated by the NSA are used as artificial abnormal samples that serve as input to a classification algorithm that learns an anomaly detection function.

Similar to the binary-valued NSA [6], the real-valued NSA [9] tries to cover the non-self space with minimum number of detectors. This is accomplished by an iterative process that updates the position of the detectors driven by two objectives: to move detectors away from self points and to keep the detectors separated in order to maximize the covering of non-self space (non-overlapping). This algorithm is shown in Figure 1.

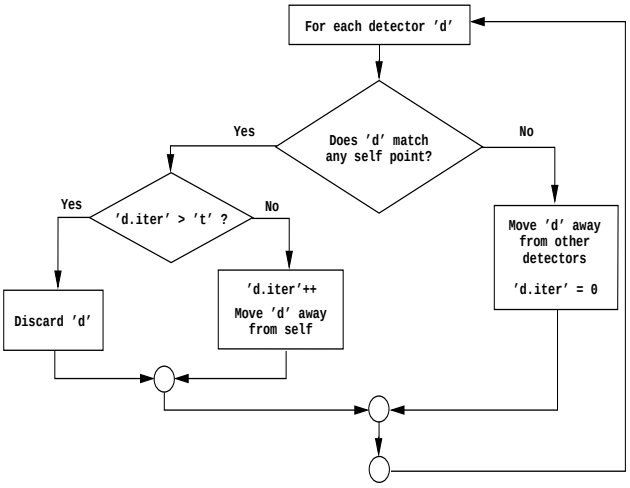


Figure 1: Illustrates an iteration of the real-valued negative selection algorithm with a flow diagram.

We used a hybrid approach by combining NSA and a neural network–multi-layer perceptron (MLP) with a hidden layer trained using back-propagation [10]. Figure 2 illustrates the basic idea of the approach. During the training stage, the input corresponds to the normal samples (feature vectors extracted from normal time series), while the NSA [9] is used to generate abnormal samples. Subsequently, the normal and abnormal samples are used to train a neural network classifier. The trained neural network corresponds to the anomaly detection function that is used during the testing phase to classify new samples as normal or abnormal.

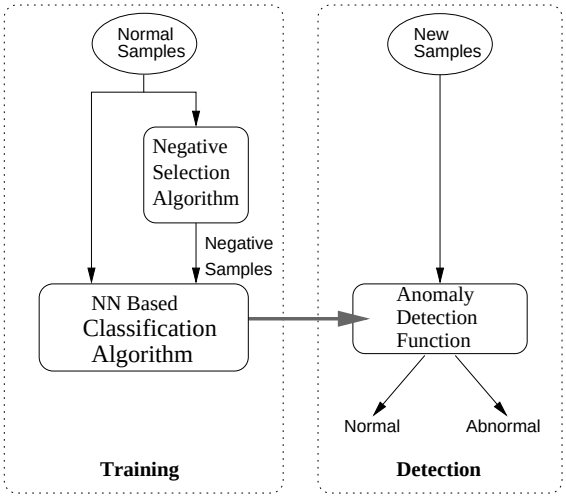


Figure 2: A process to generate an anomaly characterization function from normal samples.

3 Anomaly Detection Using Self-Organizing Maps

A self-organizing map (SOM) is a type of neural network that uses competitive learning [14, 10]. A SOM is able to capture the important features contained on the input space and provides a structural representation that preserves a topological structure. The output neurons of a SOM are organized in a one- or two-dimensional lattice. The weight vectors of these neurons represent prototypes of the input data that can be interpreted as the centroids of clusters of similar samples.

In our experiments, we used SOM to cluster the normal samples. After the network is trained, the generated clusters are used to determine if a new sample is normal or abnormal. The basic idea is: if a new sample is 'close' enough to a normal cluster it is considered normal, otherwise it is classified as abnormal.

In general, we have a distance function $dist(s, K)$ that measures how close the sample s is to the cluster, K. To determine the abnormality of a new sample, the following function is used:

$$dist(s, Normal) = \min\{dist(s, K_i) \mid K_i \in C\}$$

$$\chi_{abnormal}(s) = \begin{cases} 1 & \text{if } dist(s, Normal) \geq t \\ 0 & \text{otherwise} \end{cases},$$

where, C is the set of clusters (found by the SOM algorithm) that represents the normal sub-space. If we

think the function $dist(s, Normal)$ is a kind of membership function[1] of the abnormal subspace, the function $\chi_{abnormal}(s)$ corresponds to the crisp version of it. In this case, the value t represents a threshold that defines the boundary between the normal and abnormal classes.

In order to determine a good distance measure $dist(s, K)$, we tested three options (in all the cases w_K represents the centroid of the cluster K, neuron weights):

- **Euclidean distance**. This is the natural (or naive) choice since the SOM algorithm uses it to determine if a sample belongs to a given cluster:

$$dist(s, K) = \|s - w_K\|$$

- **Normalized distance**. The idea is to take into account the size of the cluster. Some clusters can be very sparse and others can have all the elements concentrated around the centroid. A measure of the size is the standard deviation. So, the standard deviation of the distance to the centroid of all the elements in a cluster (σ_K) is calculated and it is used to normalize the distance:

$$dist(s, K) = \frac{\|s - w_K\|}{\sigma_K}$$

- D_∞ **Minkowsky distance**. The Euclidean distance gives the same importance to all the features. So, it is possible that a sample with a non-negligible deviation in one feature will be considered as having the same overall deviation as a pattern with small deviation on many features. The D_∞ distance only takes into account the maximum of the differences for all the features:

$$dist(s, K) = \max\{|s_i - w_{K_i}| \text{ for } i = 1, \ldots, n\}$$

4 Time Series Data Set

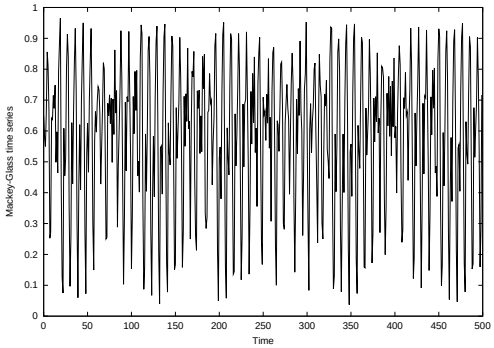
We used the Mackey-Glass equation to generate time series data. It is a non-linear, delay-differential equation whose dynamics exhibit chaotic behavior for some parameter values. The equation is:

$$\frac{dx}{dt} = \frac{ax(t-\tau)}{1 + x^c(t-\tau)} - bx(t)$$

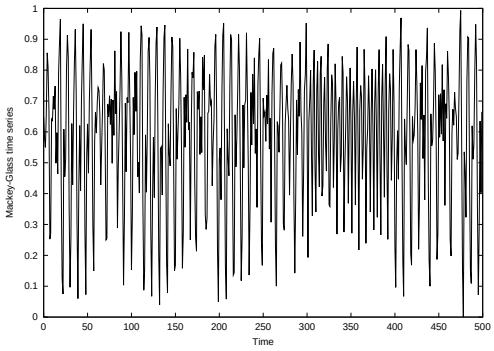
[1]Strictly speaking, this is not a membership function since it is not bounded. However, we can apply, for instance, a sigmoid function to make it bounded.

The parameters chosen were $a = 0.2$, $b = 0.1$, and $c = 10$. This set of parameters are the general choice in the literature [3, 1]. The parameter τ controls the complexity of the series dynamics. For the first experiment $\tau = 30$ was used to generate the normal samples.

The equation is solved numerically using fourth-order Runge-Kutta method (included in Matlab) with an integration step of 0.02, a sampling rate of 12, and an initial value vector with all its elements equal to 1.1. The normal samples were produced from a time series with 500 elements generated using $\tau = 30$ and discarding the first 1000 samples to eliminate the initial value effect. The resulting time series is shown in Figure 3.a.



(a) normal time series



(b) time series with an anomaly

Figure 3: Mackey-Glass series: (a) normal, using $\tau = 30$, (b) with an anomaly, $\tau = 17$ from 300 to 400.

The features are extracted using a sliding overlapping window of size n. If the time series has the values: $x_1, x_2, ..., x_m$, the feature set generated from it will be the following:

$$
\begin{array}{llll}
(x_1, & x_2, & \ldots & x_n) \\
(x_2, & x_3, & \ldots & x_{n+1}) \\
\vdots & \vdots & \vdots & \vdots \\
(x_{m-n+1} & x_{m-n+2} & \ldots & x_m)
\end{array}
$$

So, from a time series with m elements and using a sliding window of size n, we can generate (m-n+1) samples.

In order to perform the testing, we need new normal and abnormal samples. For abnormal samples, we change the parameter of the series (τ). For the preliminary experiments, we used τ = 17 (as used in [3, 1]). Figure 3.b shows an example of a time series with an abnormal segment (time 300 to 400) where the parameter τ was changed from 30 to 17.

5 Experimental Results

5.1 Experiments using SOM technique

To perform SOM experiments, we used a tool that is available on Internet (GeneCluster [19], `http://www-genome.wi.mit.edu/cancer/software/software.html`). This tool is primarily used to cluster gene expression information, however, it can be applied to any kind of data. We found the visual representation of clusters is very useful for our purpose.

For this set of experiments, we used the normal Mackey-Glass data, as plotted in Figure 3, for training. A window size of 4 was used to generate the feature vectors. Accordingly, a total number of 497 patterns were generated. The clusters generated by the application using an output grid of 6×4 neurons are shown in Figure 4. Each box shows a cluster centroid (middle curve) as well as the variations for each feature: maximum value (upper curve) and minimum value (lower curve) in the cluster. The number of samples on each cluster is also presented.

We also tested output grids with 3×4, and 8×8 neurons. In all cases, the SOM algorithm was run for 100 iterations using Gaussian neighborhood. The initial and final learning rate were 0.1 and 0.005 respectively. The initial σ value was 5 and the final was 0.2.

During testing, we applied the technique described on section 3 using the data in Figure 3.b. Figure 5 shows the anomaly detection function (i.e. $dist(s, Normal)$) for three different distance measures using an SOM with an 8×8 output layer configuration.

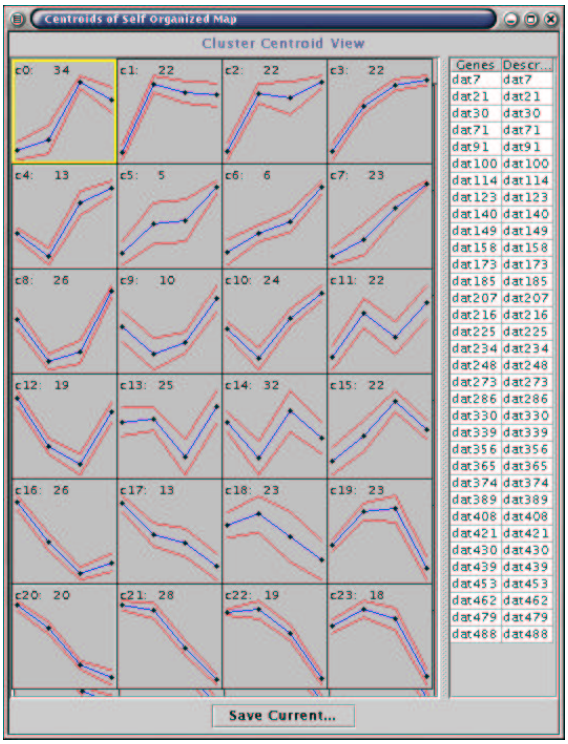
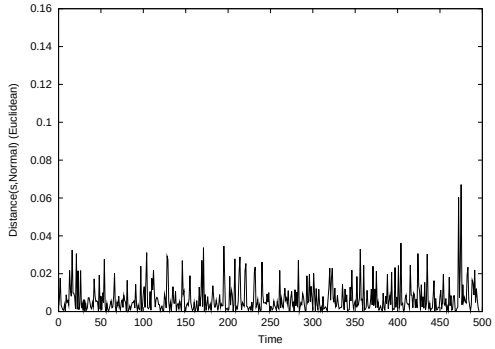


Figure 4: Clustering of the normal data produced by GeneCluster [19] (columns in right hand side are not relevant to our experiments).

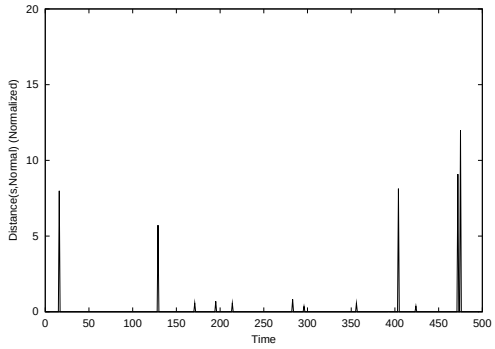
It is clear that the anomaly detection based on Euclidean distance (Figure 5.a) is not able to detect the anomalous patterns. The normalized distance does not improve either. The plots corresponding to D_∞ Minkowsky distance show an increase on the average value between the time 300 and 400 which corresponds to the anomalous section. This indicates that this distance measure is able to detect the anomalous patterns.

It is to be noted that the change on the number of output neurons reflected on the shape of the function, i.e. the more neurons on the output, the smoother the function. This is explained by the fact that more neurons imply more clusters which can approximate the normal set better.

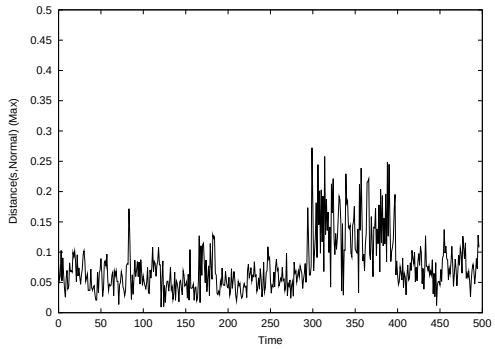
The Euclidean distance and the normalized distance assume that the clusters are spherical, that is, the distribution is the same for all directions. It seems that this is not the case, as it is evidenced in the poor performance of these distance measures. The D_∞ distance eliminates, to some degree, the interference between features and this seems to be an advantage for this specific problem. However, its main drawback is that it does not take into account the shape of the clus-



(a) Euclidean distance



(b) normalized distance



(c) D_α Minkowsky

Figure 5: Anomaly function ($dist(s, Normal)$) generated using the SOM-based technique and applied to the testing set. The net has an 8×8 output layer. Each graph represents a different distance measure: (a) Euclidean distance, (b)Normalized distance, and (c) D_∞ Minkowsky distance.

ter. Our hypothesis is that a distance measure such as Mahalanobis distance will perform much better, since it can represent ellipsoid clusters.

The anomaly function presents many peaks; in order to smooth it, a moving average technique was applied. The new output $\widehat{O_t}$ is calculated from the old output O_t using the following formula:

$$\widehat{O_t} = \frac{\sum_{i=1}^{s} O_{t-i}}{s}$$

where s is the smoothing factor and indicates the size of the averaging window. Figure 6 shows the results of the smoothing process for the anomaly function corresponding to D_∞ Minkowsky distance using $s = 10$. It is evident from the figure, how the smoothing process makes a clear boundary between the normal and the abnormal sections. As it was discussed previously, the contrast is bigger for the SOM with more output neurons (8×8). A quantitative comparison of these anomaly functions is performed in section 5.3.

5.2 Experiments using Neuro-Immune anomaly detection technique

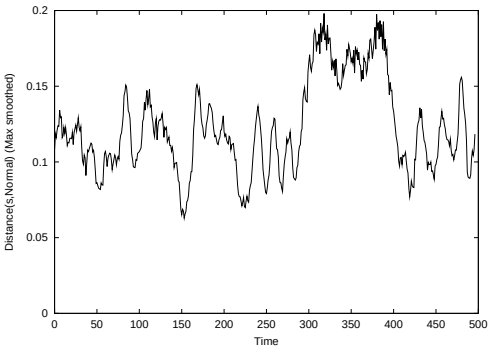
The data in Figure 3 was used to generate the training set using a window size of 4. This generated 497 normal samples that were used as input for the NSA which generated 400 abnormal samples. The normal samples were assigned an output value of 0.0 and the abnormal samples an output value of 1.0. For the classification phase, a multilayer neural network with 4 inputs, and one output neuron was used. We tested three different MLPs with 6, 12, and 16 hidden neurons respectively.

The training algorithm was back-propagation with momentum using the following parameters: learning rate 0.2, momentum 0.9, number of epochs 4000. Figure 7 shows the output of the a MLP with 16 hidden units when applied to the testing set.

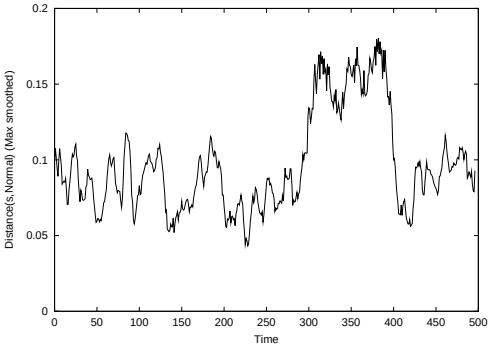
The results show that the trained MLPs are able to detect the anomalous segment present on the testing set. The output from the simplest MLP (six hidden neurons) shows more spikes. A possible explanation is that a larger number of hidden neurons allows to represent more details of the normal subspace. However, the smoothing process is able to eliminate most of them.

5.3 Comparison of the two techniques

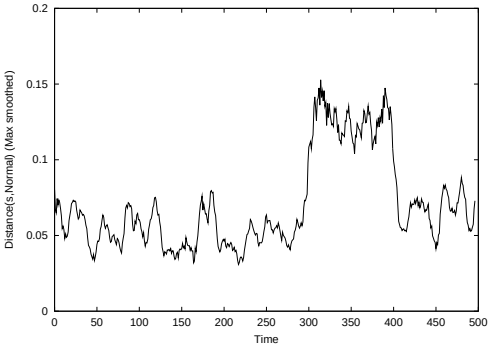
In order to compare the two techniques (SOM and neuro-immune) it is necessary to define a measure of



(a)

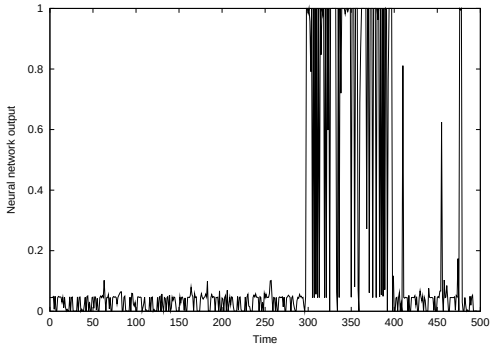


(b)

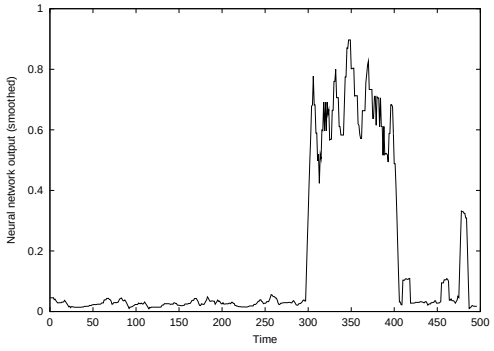


(c)

Figure 6: D_∞ Minkowsky distance anomaly function smoothed using a moving average with parameter $s = 10$. The different plots represent different topologies: (a) 3×4 neurons, (b) 6×4 neurons, and (c) 8×8 neurons.



(a) raw output (without using the smoothing function)



(b) smoothed output using $s = 10$

Figure 7: Neural network output for the testing set using 16 hidden neurons (neuro-immune technique).

accuracy for the classification. The idea is to calculate the number of true positives (TP, anomalous elements identified as anomalous), true negatives (TN, normal elements identified as normal), false positives (FP, normal elements identified as anomalous) and false negatives (FN, anomalous elements identified as normal). These values are used to calculate two measures of effectiveness:

$$\text{Detection rate} = \frac{TP}{TP + FN}$$

$$\text{False alarm rate} = \frac{FP}{TN + FP}$$

In general, we want a very high detection rate with a very low false alarm. However, there is a trade-off between these two measures. This trade-off can be shown using ROC (receiver operating characteristics) curves [18]. The sensitivity of the system is controlled

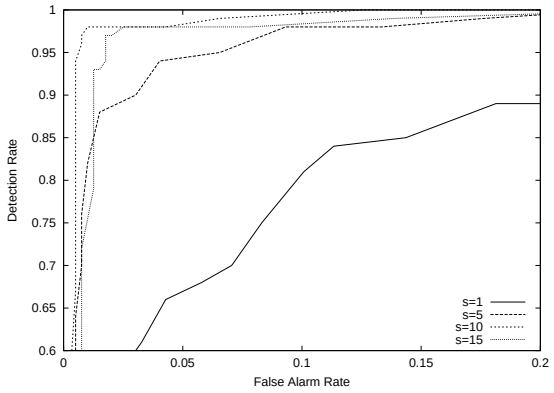
by a threshold that determines when a new sample is normal or abnormal. By varying this threshold, we can obtain different values for the detection and false alarm rates which are used to plot ROC curves.

Figure 8 shows ROC curves for SOM-based and neuro-immune anomaly detection techniques. In all cases, it is clear that the smoothing parameter (s) improves the classification accuracy. However, the SOM-based technique seems to be more sensitive to its value. This is explained by the fact that the anomaly detection function generated by this method is not as smooth as the one generated by the neuro-immune method.

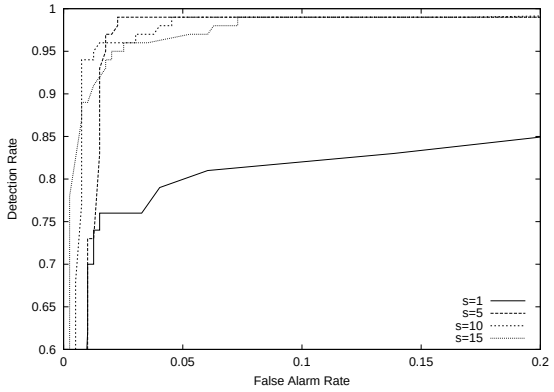
For the two methods the most complex networks generate better results. As it was explained previously, a most complex network allows a more detailed modeling of the normal subspace.

The best anomaly detection functions from the two methods are shown in Figure 9. There is no clear winner. The anomaly detection function generated by the SOM method is able to produce a very good detection rate with a low false alarm rate. But, if a small increase on the false alarm rate is allowed, the neuro-immune method is able to produce a better detection rate than the SOM method.

An important issue on anomaly detection is how to find a good threshold value that produces a detection rate with an acceptable false alarm rate. This could be very difficult if the anomaly detection function is very sensitive to this threshold. Figure 10 shows how the detection and false alarm rates change when the threshold is modified. For the neuro-immune method, the detection rate increases gradually as the threshold increases. The false alarm rate only increases at the end, producing a good range of threshold values where it is possible to have a high detection rate keeping the false alarm rate low. In the case of the SOM-based method, the detection rate changes suddenly with a small change on the threshold. The range of threshold values that can produce a good detection rate with a low false alarm rate is very small. This means, that the threshold has to be chosen very carefully and that a small variation can easily deteriorate the performance of the anomaly detection system.



(a) SOM-based method



(b) Neuro-Immune method

Figure 8: ROC curves for different values of the threshold parameter (s).

6 Conclusions

In this paper, we compared two different approaches for anomaly detection: one uses a neuro-immune technique and the other uses self-organizing maps (SOM). Their performances, from the point of view of classification accuracy, appears to be very similar. In both cases, the smoothing process (moving average) improved the classification performance significantly.

As it was expected, more complex neural networks had better performance; SOM networks were, in general, more complex than the feed-forward networks (MLP) used on the neuro-immune technique that exhibit similar performance. For instance, two networks that are compared (shown in figure 10) have $(1 + 4) \times 16 + 16 = 97$ weights (neuro-immune) and $4 \times 64 = 256$ weights (SOM) needed to be trained.

In general, the anomaly detection functions generated by the neuro-immune method were relatively smoother. This represents a clear advantage as they are less sensitive to changes on the threshold. How-

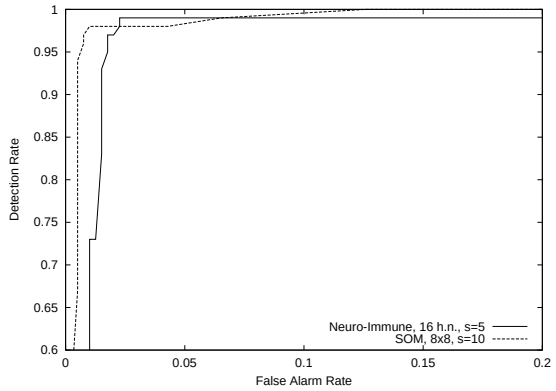
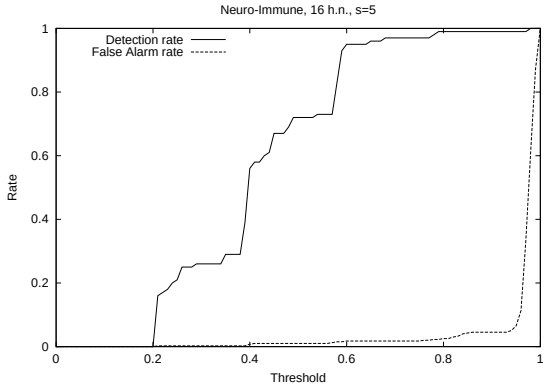


Figure 9: Best anomaly detection functions of each method.

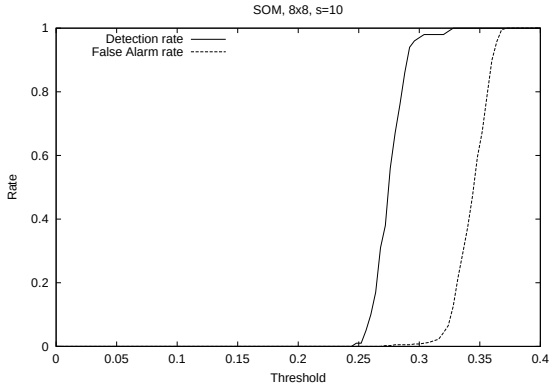
ever, there is room for improvement for the SOM method too. For instance, a distance measure that takes into account the shape of the cluster (like Mahalanobis distance) will probably improve the performance of the SOM method. So, it is necessary to test new distance measures and perform additional experiments using wide variety of data sets in order to make a fair comparison.

7 Acknowledgments

This work was funded by the Defense Advanced Research Projects Agency (no. F30602-00-2-0514) and National Science Foundation (grant no. IIS-0104251).



(a) Neuro-Immune method



(b) SOM-based method

Figure 10: Evolution of detection and false alarm rates when the threshold is modified.

References

[1] T. Caudell and D. Newman. An adaptive resonance architecture to define normality and detect novelties in time series and databases. In *IEEE World Congress on Neural Networks*, pages 166–176, Portland, Oregon, 1993.

[2] D. Dagupta and F. González. An Immunity-Based Technique to Characterize Intrusions in Computer Networks. *IEEE Transactions on Evolutionary Computation*, 6(3):1081–1088, June 2002.

[3] D. Dasgupta and S. Forrest. Novelty detection in time series data using ideas from immunology. In *Proceedings of the International Conference on Intelligent Systems*, pages 82–87, June 1996.

[4] D. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13(2):222–232, February 1987.

[5] P. D'haeseleer, S. Forrest, and P. Helman. An immunological approach to change detection: Algorithms. In *Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy*, pages 110–119, Oakland, CA, 1996. IEEE Computer Society Press.

[6] S. Forrest, A. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer. In *Proc. IEEE Symp. on Research in Security and Privacy*, pages 202–212, May 1994.

[7] K. L. Fox, R. R. Henning, J. H. Reed, and R. P. Simonian. A neural network approach towards intrusion detection. In *Proc. 13th NIST-NCSC National Computer Security Conference*, pages 125–134, 1990.

[8] F. González and D. Dasgupta. An imunogenetic technique to detect anomalies in network traffic. In *Gecco 2002: proceedings of the genetic and evolutionary computation conference*, pages 1081–1088, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[9] F. González, D. Dasgupta, and R. Kozma. Combining Negative Selection and Classification Techniques for Anomaly Detection. In *Proceedings of the Congress on Evolutionary Computation*, pages 705–710, Honolulu, HW, May 2002.

[10] S. Haykin. *Neural Networks : A Comprehensive Foundation*. Macmillan, New York, 1994.

[11] S. A. Hofmeyr and S. Forrest. Architecture for an artificial immune system. *Evolutionary Computation*, 8(4):443–473, 2000.

[12] W. H. Hsu, L. S. Auvil, W. M. Pottenger, D. Tcheng, and M. Welge. Self-organizing systems for knowledge discovery in databases. In *In Proceedings of the International Joint Conference on Neural Networks (IJCNN-99)*, Washington, DC, July 1999.

[13] E. Keogh, S. Lonardi, and B. Y. Chiu. Finding surprising patterns in a time series database in linear time and space. In *Proceedings of The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02)*, Alberta, Canada, 2002.

[14] T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 1995. (Second Extended Edition 1997).

[15] T. Lane. *Machine Learning Techniques For The Computer Security*. PhD thesis, Purdue University, West Lafayette, IN, 2000.

[16] W. Lee and S. Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, pages 26–29, San Antonio, TX, January 1998.

[17] L. Portnoy, E. Eskin, and S. J. Stolfo. Intrusion detection with unlabeled data using clustering. In *Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001)*, Philadelphia, PA, November 2001.

[18] F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Proceedings Of 15th International Conference On Machine Learning*, pages 445–453, San Francisco, Ca, 1998. Morgan Kaufmann.

[19] P Tamayo, D Slonim, J Mesirov, Q Zhu, S Kitareewan, E Dmitrovsky, ES Lander, and TR Golub. Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. *Proc Natl Acad Sci U S A*, 96(6):2907–12, March 1999.

[20] T. Y. Yoshikiyo. Fault detection by mining association rules from house-keeping data. In *Proceedings of International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2001)*, Montreal, Canada, June 2001.

An Approach to Solve Multiobjective Optimization Problems Based on an Artificial Immune System

Carlos A. Coello Coello and Nareli Cruz Cortés
CINVESTAV-IPN (Evolutionary Computation Group)
Departamento de Ingeniería Eléctrica/Sección de Computación
Av. IPN No. 2508, Col. San Pedro Zacatenco, México, D. F. 07300
ccoello@cs.cinvestav.mx, nareli@computacion.cs.cinvestav.mx

Abstract

In this paper, we propose an algorithm to solve multiobjective optimization problems (either constrained or unconstrained) using the clonal selection principle. Our approach is compared with respect to another algorithm that is representative of the state-of-the-art in evolutionary multiobjective optimization. For our comparative study, two metrics are adopted and graphical comparisons with respect to the true Pareto front of each problem are also included. Results indicate that the proposed approach is very promising.

1 Introduction

The immune system is one of the most important biological mechanisms humans possess since our own life depends on it. In recent years, several researchers have developed computational models of the immune system that attempt to capture some of their most remarkable features such as its self-organizing capability [11, 9].

From the information processing perspective, the immune system can be seen as a parallel and distributed adaptive system [10, 3]. It is capable of learning, it uses memory and is able of associative retrieval of information in recognition and classification tasks. Particularly, it learns to recognize patterns, it remembers patterns that it has been shown in the past and its global behavior is an emergent property of many local interactions [3]. All these features of the immune system provide, in consequence, great robustness, fault tolerance, dynamism and adaptability [9]. These are the properties of the immune system that mainly attract researchers to try to emulate it in a computer.

In this paper, we propose an approach to solve multiobjective optimization problems (either with or without constraints) based on the clonal selection principle.

2 The Immune System

The main goal of the immune system is to protect the human body from the attack of foreign (harmful) organisms. The immune system is capable of distinguishing between the normal components of our organism and the foreign material that can cause us harm (e.g., bacteria). These foreign organisms are called *antigens*.

The molecules called *antibodies* play the main role on the immune system response. The immune response is specific to a certain foreign organism (antigen). When an antigen is detected, those antibodies that best recognize an antigen will proliferate by cloning. This proccess is called *clonal selection principle* [4].

The new cloned cells undergo high rate mutations or *hypermutation* in order to increase their receptor population (called repertoire). These mutations experienced by the clones are proportional to their affinity to the antigen.

The highest affinity antibodies experiment the lowest mutation rates, whereas the lowest affinity antibodies have high mutation rates. After this mutation process ends, some clones could be dangerous for the body and should therefore be eliminated.

After these clonation and hypermutation processes finish, the immune system has improved the antibodies' affinity, which results on the antigen neutralization and elimination.

At this point, the immune system must return to its normal conditions, eliminating the excedent cells. However, some cells remain circulating throughout the body as memory cells. When the immune system is later attacked by the same type of antigen (or a sim-

ilar one), these memory cells are activated, presenting a better and more efficient response. This second encounter with the same antigen is called *secondary response*.

The algorithm proposed in this paper is based on the clonal selection principle previously described.

3 Multiobjective Optimization

Multiobjective optimization (also called multicriteria optimization, multiperformance or vector optimization) can be defined as the problem of finding [15]:

> a vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term "optimize" means finding such a solution which would give the values of all the objective functions acceptable to the designer.

Formally, we can state the general multiobjective optimization problem (MOP) as follows:

Definition 1 (General MOP): *Find the vector* $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ *which will satisfy the* m *inequality constraints:*

$$g_i(\vec{x}) \geq 0 \quad i = 1, 2, \ldots, m \tag{1}$$

the p *equality constraints*

$$h_i(\vec{x}) = 0 \quad i = 1, 2, \ldots, p \tag{2}$$

and optimizes the vector function

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T \tag{3}$$

where $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ *is the vector of decision variables.* □

In other words, we wish to determine from among the set $\mathcal{F}$ of all numbers which satisfy (1) and (2) the particular set $x_1^*, x_2^*, \ldots, x_n^*$ which yields the optimum values of all the k objective functions of the problem.

Another important concept is that of Pareto optimality, which was stated by Vilfredo Pareto in the XIX century [16], and constitutes by itself the origin of research in multiobjective optimization:

Definition 2 (Pareto Optimality:): *We say that* $\vec{x}^* \in \mathcal{F}$*, is* **Pareto optimal** *if for every* $\vec{x} \in \Omega$ *and* $I = \{1, 2, \ldots, k\}$ *either,*

$$\bigwedge_{i \in I} (f_i(\vec{x}) = f_i(\vec{x}^*)) \tag{4}$$

or, there is at least one $i \in I$ *such that (assuming maximization)*

$$f_i(\vec{x}) \leq f_i(\vec{x}^*) \tag{5}$$

□

In words, this definition says that $\vec{x}^*$ is Pareto optimal if there exists no feasible vector $\vec{x}$ which would increase some criterion without causing a simultaneous decrement in at least one other criterion.

Pareto optimal solutions are also termed non-inferior, admissible, or efficient solutions [2]; their corresponding vectors are termed nondominated. These solutions may have no clearly apparent relationship besides their membership in the Pareto optimal set. This is the set of all solutions whose corresponding vectors are nondominated with respect to all other comparison vectors. When plotted in objective space, the nondominated vectors are collectively known as the Pareto front.

4 The Proposed Approach

As indicated before, our algorithm is based on the *clonal selection principle*, modeling the fact that only the highest affinity antibodies to the antigens will proliferate. Our algorithm uses the concept of Pareto dominance to generate nondominated vectors. Also, an external (or secondary) memory is used to store nondominated vectors found along the evolutionary process, in order to move towards the true Pareto front over time (this can be seen as a form of elitism in evolutionary multiobjective optimization [2]).

4.1 The Algorithm

Our algorithm is the following:

1. Generate randomly the initial population.
2. Initialize the secondary memory so that it is empty.

3. Determine for each individual in the population, if it is (Pareto) dominated or not. For constrained problems, determine if an individual is feasible or not.

4. Split the population into antigens and antibodies. The division criterion is Pareto dominance (i.e., nondominated individuals are the antigens and dominated individuals are the antibodies). In constrained problems, feasible individuals are antigens, too. Note that either of the two criteria (Pareto dominance or feasibility) is sufficient for an individual to be considered an antigen. However, to guide the search properly, we distinguish between "very good" (or ideal) antigens and those which are only "good". For that sake, we assign a weight (w) to each antigen according to the following rules:

   - $w = 4$ for nondominated and feasible antigens (the best ones).
   - $w = 3$ for nondominated antigens (even if infeasible).
   - $w = 2$ for feasible antigens (even if they are dominated).

   Note that in the previous rules, Pareto dominance is given more importance than feasibility. These values were arbitrarily adopted to give more or less importance to each of the cases previously indicated. Note however, that the same values are adopted in all the examples presented in this paper. Also, note that in unconstrained problems, all nondominated individuals are made antigens with a $w = 2$.

5. Copy the antigens (with $w = 4$ for constrained problems and with $w = 2$ for unconstrained problems) to the secondary memory.

6. Select an antigen (regardless of its weight) at random.

7. Assign a fitness value to each of the antibodies according to their matching value (Z) with respect to the antigen (randomly) chosen from the previous step (see Figure 1). Note that a new antigen is randomly selected for each antibody.

8. Select the Q fittest antibodies from the antibodies pool where the fitness criterion is defined by the value of Z.

9. Create a number N of copies of the antibodies selected.

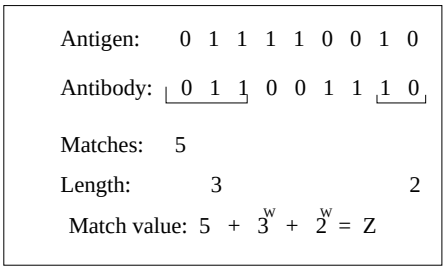


Figure 1: Matching measure between an antigen and an antibody. The weights w are used to increase the value of Z when an antibody matches a highly desirable antigen (i.e., nondominated and feasible).

10. Assign a mutation rate (MR) to each clone, according to their similarity with an antigen randomly chosen. The higher the similarity the lower the mutation rate, and viceversa.

11. Apply mutation rate MR to each clone.

12. The new population is formed by the union of the original antibodies and their clones.

13. The population size is returned to its original value, allowing the nondominated individuals (and the feasible ones if dealing with a constrained problem) survive.

14. Go back to step 3 until convergence occurs or after reaching a certain (predetermined) number of iterations.

The antigen-antibody matching measure (Z) adopted in this paper is adapted from Farmer's proposal [7]. This matching measure counts the number of matching bits of the two strings compared as well as the number of consecutive matching bits. For example, if we have three contiguous similarities on the strings we add a value of 3 raised to its w value to the total matching measure (see figure 1).

Note that this algorithm is not really a genetic algorithm since no sexual recombination takes place. Instead, only a clonation of individuals is used to generate the new population of the algorithm.

4.2 Secondary Memory

We use a secondary or external memory as an elitist mechanism in order to maintain the best solutions found along the process. The individuals stored in this memory are all nondominated not only with respect to each other but also with respect to all of the previous individuals who attempted to enter the external

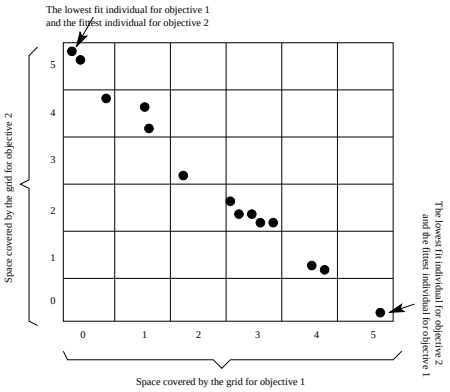


Figure 2: An adaptive grid to hadle the secondary memory

memory. Therefore, the external memory stores our approximation to the true Pareto front of the problem.

In order to enforce a uniform distribution of nondominated solutions that cover the entire Pareto front of a problem, we use the adaptive grid proposed by Knowles and Corne [13] (see Figure 2).

Ideally, the size of the external memory should be infinite. However, since this is not possible in practice, we must set a limit to the number of nondominated solutions that we want to store in this secondary memory. By enforcing this limit, our external memory will get full at some point even if there are more nondominated individuals wishing to enter. When this happens, we use an additional criterion to allow a nondominated individual to enter the external memory: region density (i.e., individuals belonging to less densely populated regions are given preference).

The algorithm for the implementation of the adaptive grid is the following:

1. Divide objective function space according to the number of subdivisions set by the user.
2. For each individual in the external memory, determine the cell to which it belongs.
3. If the external memory is full, then determine which is the most crowded cell.
4. To determine if a certain antigen is allowed to enter the external memory, do the following:
   - If it belongs to the most crowded cell, then it is not allowed to enter.
   - Otherwise, the individual is allowed to enter. For that sake, we eliminate a (randomly chosen) individual that belongs to the most crowded cell in order to have an available slot for the antigen.

5 Experiments

In order to validate our approach, we used several test functions reported in the standard evolutionary multiobjective optimization literature [5, 20, 2]. In each case, we generated the true Pareto front of the problem (i.e., the solution that we wished to achieve) by enumeration using parallel processing techniques. Then, we plotted the Pareto front generated by our algorithm, which we call the multiobjective immune system algorithm (MISA). The results indicated below were found using the following parameters: Maximum number of iterations = 150, population size = 70, clonation rate = 0.8, number of clones = 15, size of the external memory = 100. The above parameters produce a total of 138,000 fitness function evaluations.

MISA was compared against the micro-genetic algorithm for multiobjective optimization, which was recently proposed [1]. This algorithm is representative of the state-of-the-art in evolutionary multiobjective optimization and has been found to produce similar or better results than the NSGA-II [6] and PAES [13].

To allow a fair comparison, the micro-GA performed the same number of fitness function evaluations as MISA.

Despite the graphical comparisons performed, the two following metrics were adopted to compare our results:

- **Two Set Coverage (SC)**: This metric was proposed in [22], and it can be termed *relative coverage comparison of two sets*. Consider $X', X'' \subseteq X'$ as two sets of phenotype decision vectors. SC is defined as the mapping of the order pair (X', X'') to the interval $[0, 1]$.

$$SC(X', X'') \triangleq \frac{|\{a'' \epsilon X''; \exists a' \epsilon X' : a' \succeq a''\}|}{|X''|} \quad (6)$$

  If all points in X' dominate or are equal to all points in X'', then by definition $SC = 1$. $SC = 0$ implies the opposite. In general, $SC(X', X'')$ and $SC(X'', X')$ both have to be considered due to set intersections not being empty. Of course, this metric can be used for both spaces (objective function or decision variable space), but in this case we applied it in objective function space. The advantage of this metric is that it is easy to calculate and provides a relative comparison based

upon dominance numbers between generations or algorithms.

- **Spacing (S)**: This metric was proposed by Schott [18] as a way of measuring the range (distance) variance of neighboring vectors in the Pareto front known. This metric is defined as:

$$S \triangleq \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\overline{d}-d_i)^2}\ , \qquad (7)$$

where $d_i = \min_j(\mid f_1^i(\vec{x}) - f_1^j(\vec{x}) \mid + \mid f_2^i(\vec{x}) - f_2^j(\vec{x}) \mid)$, $i,j = 1,\ldots,n$, $\overline{d}$ is the mean of all d_i, and n is the number of vectors in the Pareto front found by the algorithm being evaluated. A value of zero for this metric indicates all the nondominated solutions found are equidistantly spaced.

The parameters used by the micro-GA for the experiments reported below are the following: maximum number of generations = 8400, population size = 4, number of grid subdivisions = 25, memory size = 50, crossover rate = 0.8, number of iterations to achieve nominal convergence = 4, size of the external memory = 100. We the previous parameters, the micro-GA performs a total of 138,000 fitness function evaluations.

Example 1

Minimize: $F = (f_1(x,y), f_2(x,y))$, where

$$\begin{aligned} f_1(x,y) &= x, \\ f_2(x,y) &= (1+10y)* \\ &\quad [1-(\frac{x}{1+10y})^{\alpha} - \frac{x}{1+10y}\sin(2\pi q x)] \end{aligned}$$

and $0 \le x,y \le 1$, $q = 4$, $\alpha = 2$.

The comparison of results between the true Pareto front of this example and the Pareto front produced by MISA is shown in Figure 3. Note that the Pareto front is disconnected (it consists of four Pareto curves). In this case: $SC(MISA, micro-GA) = 0.304$ and $SC(micro-GA, MISA) = 0.29$. This indicates a very similar behavior from both algorithms and we can say that there is a tie among the final nondominated solutions produced by the two algorithms. In terms of spacing, the results are presented in Table 1. Note that the average results of MISA are better than those of the micro-GA.

Table 1: Spacing for example 1

	best	average	worst	std.dev.
MISA	0.008853	0.114692	0.62904	0.175955
micro-GA	0.007773	0.177104	0.991838	0.319061

Table 2: Spacing for example 2

	best	average	worst	std.dev.
MISA	0.008853	0.107427	0.209062	0.054843
micro-GA	0.04119	0.1446	1.197458	0.253813

Example 2

Our second example is a two-objective optimization problem proposed by Schaffer [17] that has been used by several researchers [19]:

$$\text{Minimize } f_1(x) = \begin{cases} -x & \text{if } x \le 1 \\ -2+x & \text{if } 1 < x \le 3 \\ 4-x & \text{if } 3 < x \le 4 \\ -4+x & \text{if } x > 4 \end{cases} \qquad (8)$$

$$\text{Minimize } f_2(x) = (x-5)^2 \qquad (9)$$

and $-5 \le x \le 10$.

The comparison of results between the true Pareto front of this example and the Pareto front produced by MISA is shown in Figure 4. In this case: $SC(MISA, micro-GA) = 0.487$ and $SC(micro-GA, MISA) = 0.56$. As in the previous example, these values indicate a very similar behavior from both algorithms and we can say that there is a tie among the final nondominated solutions produced by the two algorithms. In terms of spacing, the results are shown in Table 2. Note again that the average results of MISA are better than those of the micro-GA.

Example 3

The third example is the three-objective function problem proposed by Viennet [21]:

Minimize: $F = (f_1(x,y), f_2(x,y), f_3(x,y))$

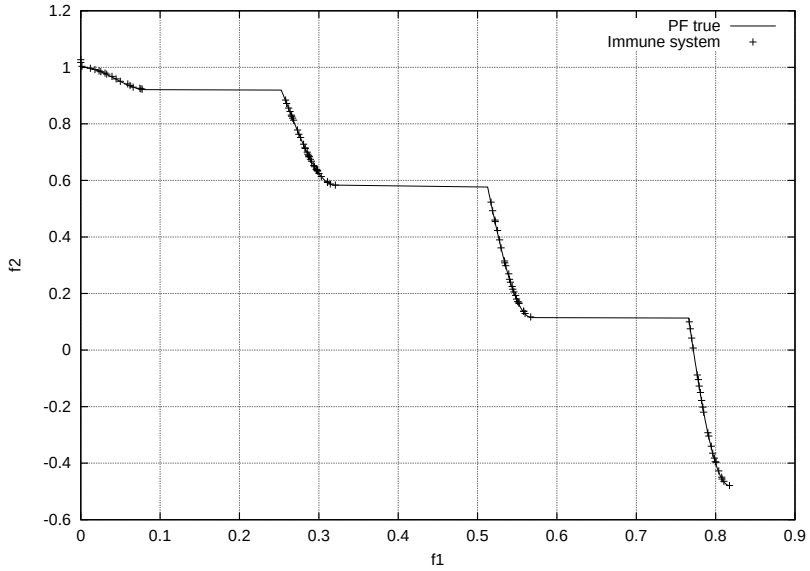


Figure 3: Comparison of results for the first example. The true Pareto front is shown as a continuous line (note that the horizontal segments are NOT part of the Pareto front and are shown only to facilitate drawing the front) and the Pareto front found by MISA is shown as crosses.

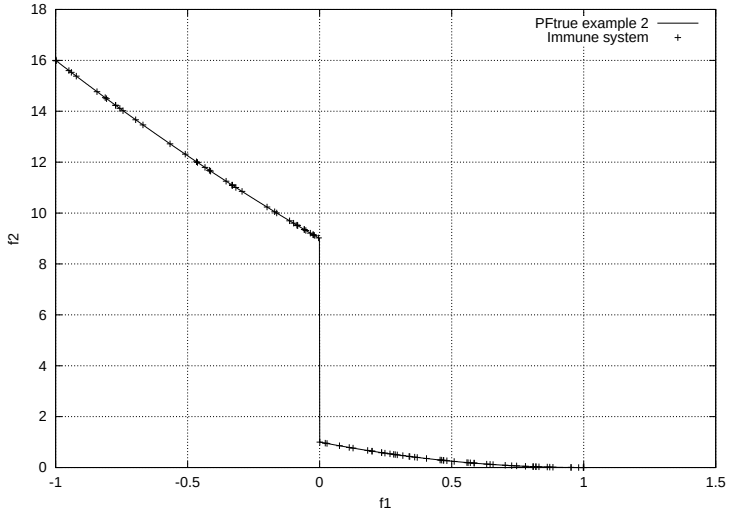


Figure 4: Comparison of results for the second test function. The true Pareto front of the problem is shown as a continuous line (note that the vertical segment is NOT part of the Pareto front and is shown only to facilitate drawing the front) and the Pareto front found by MISA is shown as crosses.

where:

$$
\begin{aligned}
f_1(x,y) &= \frac{(x-2)^2}{2} + \frac{(y+1)^2}{13} + 3, \\
f_2(x,y) &= \frac{(x+y-3)^2}{175} + \frac{(2y-x)^2}{17} - 13, \\
f_3(x,y) &= \frac{(3x-2y+4)^2}{8} + \frac{(x-y+1)^2}{27} \\
&\quad +15
\end{aligned}
$$

and: $-4 \leq x, y \leq 4$, $y < -4x + 4$, $x > -1$, $y > x - 2$.

The comparison of results between the true Pareto front of this example and the Pareto front produced by MISA is shown in Figure 5. In this case: $SC(MISA, micro - GA) = 0.673$ and $SC(micro - GA, MISA) = 0.605$. As in the previous example, these values indicate a very similar behavior from both algorithms and we can say that there is a tie among the final nondominated solutions produced by the two algorithms. In terms of spacing, the results are shown in Table 3. Note that the average results of the micro-GA are better than those of MISA. In this case, MISA had a poorer performance in terms of uniform distribution than the micro-GA.

Example 4

The fourth example was proposed by Kita [12]:

Maximize $F = (f_1(x,y), f_2(x,y))$

where:

$$
\begin{aligned}
f_1(x,y) &= -x^2 + y, \\
f_2(x,y) &= \frac{1}{2}x + y + 1
\end{aligned}
$$

$x, y \geq 0$, $0 \geq \frac{1}{6}x + y - \frac{13}{2}$, $0 \geq \frac{1}{2}x + y - \frac{15}{2}$, $0 \geq 5x + y - 30$.

The comparison of results between the true Pareto front of this example and the Pareto front produced by MISA is shown in Figure 6. In this case: $SC(MISA, micro - GA) = 1.00$ and $SC(micro - GA, MISA) = 0.145$. In this case, MISA produced solutions that clearly dominated or were equal to those generated by the micro-GA (therefore the value of 1.0). This clearly indicates a better behavior of MISA. In terms of spacing, the results are shown in Table 4. In terms of this metric, the average results of the micro-GA are better than those of MISA. Note however, that since the solutions generated by the micro-GA are covered (i.e., dominated) by those produced by MISA, the fact that these solutions have a more uniform distribution is less relevant, since these solutions are poorer than those generated by MISA.

Table 4: Spacing for example 4

	best	average	worst	std.dev.
MISA	0.141532	0.518706	1.145541	0.349627
micro-GA	0.039568	0.115826	0.830159	0.180039

Example 5

Our fifth example is a two-objective optimization problem defined by Kursawe [14]:

$$
\text{Minimize } f_1(\vec{x}) = \sum_{i=1}^{n-1} \left(-10 \exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2} \right) \right) \tag{10}
$$

$$
\text{Minimize } f_2(\vec{x}) = \sum_{i=1}^{n} \left(|x_i|^{0.8} + 5\sin(x_i)^3 \right) \tag{11}
$$

where:

$$
-5 \leq x_1, x_2, x_3 \leq 5 \tag{12}
$$

The comparison of results between the true Pareto front of this example and the Pareto front produced by MISA is shown in Figure 7. In this case: $SC(MISA, micro - GA) = 0.3490$ and $SC(micro - GA, MISA) = 0.96$. In this case, the micro-GA produced solutions that clearly dominated or were equal to those generated by MISA (therefore the value very close to 1.0). This clearly indicates a better behavior of the micro-GA. In terms of spacing, the results are shown in Table 5. Note that the average results of MISA are better than those of the micro-GA. Note however, that since the solutions generated by MISA are covered (i.e., dominated) by those produced by the micro-GA, the fact that these solutions have a more uniform distribution is less relevant, since these solutions are poorer than those generated by the micro-GA.

Summarizing, we can see that our approach has a very competitive behavior with respect to the micro-GA when dealing with unconstrained test functions. However, in constrained test functions is not as competitive (in general), but the results are still acceptable as

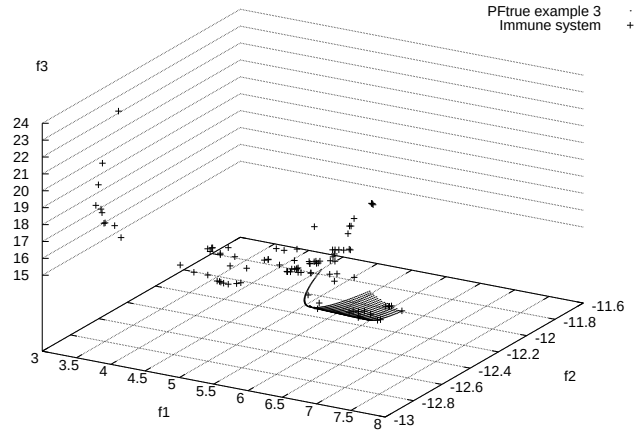


Figure 5: Comparison of results for the third test function. The true Pareto front of the problem is shown as dots and the Pareto front found by MISA is shown as crosses.

Table 3: Spacing for example 3

	best	average	worst	std.dev.
MISA	0.382708	0.515023	0.632426	0.057835
micro-GA	0.270519	0.294236	0.315999	0.012565

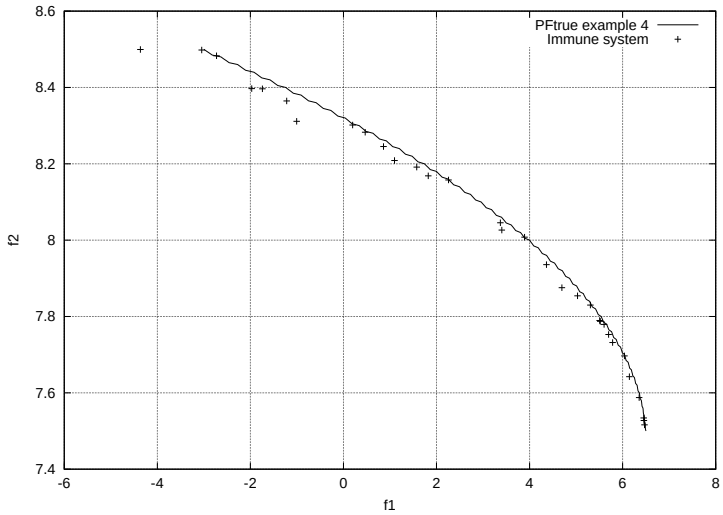


Figure 6: Comparison of results for the fourth test function. The true Pareto front of the problem is shown as a continuous line and the Pareto front found by MISA is shown as crosses.

Table 5: Spacing for example 5

	best	average	worst	std.dev.
MISA	2.008484	2.382588	3.201155	0.292547
micro-GA	2.945237	3.299231	3.905389	0.353365

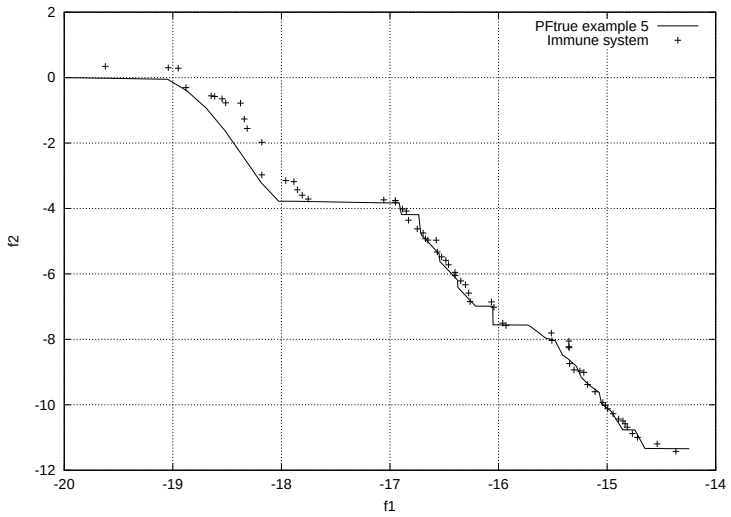


Figure 7: Comparison of results for the fifth test function. The true Pareto front of the problem is shown as a continuous line (note that the horizontal segment is NOT part of the Pareto front and is shown only to facilitate drawing the front) and the Pareto front found by MISA is shown as crosses.

can be seen in the corresponding graphs. Nevertheless, further improvements are required so that MISA can incorporate constraints more efficiently into its fitness function.

6 Conclusions and Future Work

We have presented a new multiobjective optimization algorithm based on the *clonal selection principle*. The approach seems promising and is able to produce results similar or better than those generated by an algorithm that represents the state-of-the-art in evolutionary multiobjective optimization when dealing with unconstrained test functions. However, the algorithm still requires further improvements so that it can handle constraints more efficiently. Such work is currently under way.

Additionally, we will be performing direct comparisons with other evolutionary multiobjective optimization techniques such as PAES [13], the NSGA-II [6] and MOGA [8] with elitism. In such comparative study, additional metrics will be implemented.

Our goal is to produce a highly competitive algorithm (based on the artificial immune system) that represents a viable alternative to solve multiobjective optimization problems of any kind (either constrained or unconstrained).

Acknowledgements

We thank the comments of the anonymous reviewers that greatly helped us to improve the contents of this paper. The first author gratefully acknowledges support from CONACyT through project 34201-A. The second author acknowledges support from CONACyT through a scholarship to pursue graduate studies at the Computer Science Section of the Electrical Engineering Department at CINVESTAV-IPN.

References

[1] Carlos A. Coello Coello and Gregorio Toscano Pulido. Multiobjective Optimization using a Micro-Genetic Algorithm. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 274–282, San Francisco, California, 2001. Morgan Kaufmann Publishers.

[2] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.

[3] Dipankar Dasgupta, editor. *Artificial Immune Systems and Their Applications*. Springer-Verlag, Berlin, 1999.

[4] Leandro Nunes de Castro and Fernando José Von Zuben. Artificial Immune Systems: Part I - Basic Theory and Applications. Technical Report TR-DCA 01/99, FEEC/UNICAMP, Brazil, December 1999.

[5] Kalyanmoy Deb. Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. *Evolutionary Computation*, 7(3):205–230, Fall 1999.

[6] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[7] J. D. Farmer, N. H. Packard, and A. S. Perelson. The Immune System, Adaptation, and Machine Learning. *Physica D*, 22:187–204, 1986.

[8] Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. Morgan Kauffman Publishers.

[9] Stephanie Forrest and Steven A. Hofmeyr. Immunology as Information Processing. In L.A. Segel and I. Cohen, editors, *Design Principles for the Immune System and Other Distributed Autonomous Systems*, Santa Fe Institute Studies in the Sciences of Complexity, pages 361–387. Oxford University Press, 2000.

[10] Steven A. Frank. *The Design of Natural and Artificial Adaptive Systems*. Academic Press, New York, 1996.

[11] John E. Hunt and Denise E. Cooke. An adaptative, distributed learning systems based on the immune system. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernatics*, pages 2494–2499, 1995.

[12] Hajime Kita, Yasuyuki Yabumoto, Naoki Mori, and Yoshikazu Nishikawa. Multi-Objective Optimization by Means of the Thermodynamical Genetic Algorithm. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature—PPSN IV*, Lecture Notes in Computer Science, pages 504–512, Berlin, Germany, September 1996. Springer-Verlag.

[13] Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[14] Frank Kursawe. A Variant of Evolution Strategies for Vector Optimization. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop, PPSN I*, volume 496 of *Lecture Notes in Computer Science*, pages 193–197, Berlin, Germany, oct 1991. Springer-Verlag.

[15] Andrzej Osyczka. Multicriteria optimization for engineering design. In John S. Gero, editor, *Design Optimization*, pages 193–227. Academic Press, 1985.

[16] Vilfredo Pareto. *Cours D'Economie Politique*, volume I and II. F. Rouge, Lausanne, 1896.

[17] J. David Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, 1984.

[18] Jason R. Schott. Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1995.

[19] N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, Fall 1994.

[20] David A. Van Veldhuizen and Gary B. Lamont. MOEA Test Suite Generation, Design & Use. In Annie S. Wu, editor, *Proceedings of the 1999 Genetic and Evolutionary Computation Conference. Workshop Program*, pages 113–114, Orlando, Florida, July 1999.

[21] Rémy Viennet, Christian Fontiex, and Ivan Marc. New Multicriteria Optimization Method Based on the Use of a Diploid Genetic Algorithm: Example of an Industrial Problem. In J. M. Alliot, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Proceedings of Artificial Evolution (European Conference, selected papers)*, pages 120–127, Brest, France, September 1995. Springer-Verlag.

[22] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.

Immunocomputing for Complex Interval Objects

Svetlana P. Sokolova, Ludmila A. Sokolova

St. Petersburg Institute for Informatics and Automation,
Russian Academy of Sciences,
14-line 39, St. Petersburg, 199178, Russia

Abstract

This paper provides a further development of the Immunocomputing (IC) approach to the class of complex objects with parameter uncertainty of the interval type. By using the rules and nomenclature of interval mathematics the singular value decomposition (SVD) of interval matrices, procedures for supervised learning, unsupervised learning, classification and presentation of the results of research in IC shape space have been further developed. This paper includes examples of Specific Interval Artificial Immune Systems for Surveillance of the Plague and Security Systems.

1 Introduction

A new computational technique, called the Artificial Immune Systems (AIS), base on the principles established for the immune system, can learn new information, recall previously learned information, and perform pattern recognition in a highly decentralized fashion. AISs offer powerful and robust information processing capabilities for solving complex problems. A rigorous mathematical basis of AIS based on the biological prototype of immune network and the notations of formal protein and formal immune network have been proposed (Tarakanov, 2000, 2001, 2002). These mathematical models have been reffered, as *formal immune system*, or *immunocomputing* (IC). This approach has already been applied in several specific problems, such as monitoring of the natural plague foci (Tarakanov, Sokolova, 2000), intelligent security systems (Sokolova, 2000), etc.

Development of specific applications has shown the exsistence of extensive groups of biological, economical and natural factors, for which the measured values of the state vectors are often known with interval uncertainty. This uncertainty has a non-static nature, or there is not enough information in order to refer it to one of the group of random processes. The parametric uncertainties are characterized by a relationship of the true values of the parameters of an object to intervals with known boundaries. Actually the interval model of a system mirrors a real situation with the information on values of its parameters, when a priori only the boundaries of intervals are known. Therefore, using the rules and nomenclature of interval mathematics we can represent it as mathematical models. The interval space has the following mathematical characteristics: the incompleteness of the algebraic and the ordinal structure; the lack of a rigorous distributivity (Alefeld, 1983, Neumaier, 1990, Shary, 1985). These characteristics, make the resolution of interval space uncertainties highly complicated. Its concept, as a rule, is NP-hard. Consequently, computer solutions of inner and outer estimation of these concepts in the interval analysis are necessary to simplify the process.

This paper provides a further development of the Immunocomputing approach on the class of objects with parameter uncertainty of interval type. The concept of a solution set of singular values for an interval matrix, singular value decomposition of interval matrices, procedures for supervised learning, unsupervised learning, classification and presentation of the results of research into characteristics of interval objects on shape space are developed. This includes examples of Specific Interval Artificial Immune Systems for Surveillance the Plague and Security Systems.

2 Mathematical Basis

2.1 General Approach

We will use the following notations: R^m, $R^{m\times m}$ – correspodingly the space of real vectors U with m components and the space of real $m\times m$ matrices A. IR^m is the set of *interval vectors [U]*

$$[U] = [U_-, U_+] = \{U\in R^m \mid U_- \le U \le U_+\},\ U_- \le U_+$$

with m components and $IR^{m\times m}$ is the set of *interval matrices*

$$[A]=[A_-, A_+] = \{A\in R^{m\times m} \mid A_- \le A \le A_+\},\ A_- \le A_+,$$

all inequalities are defined componentwise. Real interval coefficients of *[A]* are: $([a_{ij}], i,j=1,\ ,m) =([a_{ij}^-, a_{ij}^+])$, a_{ij}^-, a_{ij}^+ - lower and upper bounds of intervals. For $U \subseteq R^m$ *the intervall hull* (U) is defined by

$$(U) = \cap \{[v]\in IR^m \mid U\subseteq [v]\}.$$

Let a real valued interval matrix $[A]\in IR^{m\times n}$ is given. We shall understand the interval matrix $[A]$ as a set of real valued matrices of dimension $m \times n$, for which

$$\{A| (\forall A \in [A])\}.$$

According to (Tarakanov, 2000) any unit vector U with m real-valued components

$$U= [u_1, \ldots, u_m]^T,\ UU^T = 1,$$

can be considered as a special kind of Formal Peptide (FP) with $m–1$ links. Binding energy between any pair of such FPs: $\{U, V\}$, of the dimensions $(m\times1)$and $(n\times1)$, correspondingly, for given $[A]\in IR^{m\times n}$ is been defined by a interval bilinear form:

$$[\omega]=–U^T[A]V, \qquad (1)$$

where $[\omega]\in IR^1$, $[\omega] = [\omega_-, \omega_+] = \{\omega \in R^1 \mid \omega_- \le \omega \le \omega_+\}$, $\omega_- \le \omega_+$. Binding energy $\omega \in R^1$ for $\forall A \in [A]$ is been defined by a bilinear form:

$$\omega = – U^TAV, \qquad (2)$$

where $A\in R^{n\times m}$. As it is known that extreme values of the bilinear form (2) are determined by the so-called Singular Value Decomposition (SVD) of the matrix $A\in R^{n\times m}$:

$$A = s_1U_1V_1^T+ s_2U_2V_2^T + \ldots..+s_pU_pV_p^T, \qquad (3)$$

where s_i, $i = 1,\ldots, p$ - are singular values of the matrix $\forall A \in [A]$, U_i, V_i, $i = 1,\ldots, p$ - its left and right singular vectors, p - rank of the matrix A. These singular values and vectors satisfy the following bindings:

$$s_1 \ge s_2 \ge \ldots s_p \ge 0,\ s_i = U_i^TAV_i,\ V_i^TV_i = 1,\ U_i^TU_i = 1,\ i = 1,\ldots,p.$$

Let $[A] \in IR^{n\times m}$. Given an interval matrix $[A] \in IR^{n\times m}$, we look for interval quantities $[s_1] \in IR$, $[U]\in IR^m$, $[V]\in IR^n$ satisfying the following properties:

- $[s_1]$ contains an maximal singular value of each matrix $\forall A \in [A]$;
- for each of these singular values $[U]\in IR^m$, $[V]\in IR^n$ contain at least one corresponding left and right singular vectors.

Below we will consider two approaches of computing the singular value of the interval matrix $[A]$:center approach and adaptive approach.

2.2 Center Approach

Assume, that the interval matrix $[A]$ can be presented as $[A] =[A^C–\Delta, A^C+\Delta]$, $A^C = mid\ ([A])$, $mid[a_{ij}] = a_{ij}^- +0.5(a_{ij}^- – a_{ij}^+)$ is the midpoint of $[a_{ij}]$, $A^C \in R^{n\times m}$, $\Delta \ge 0$, $\Delta \in R^{n\times m}$. The midpoint of the interval matrices is defined componentwise. Computing the singular values of the interval matrix $[A]$ can be done by calculating bounds on the eigenvalues of the symmetric interval matrix $[B]= [A]^T[A]$ (Deif, 1991). However, such an approach overestimates the true bounds. The latter are better obtained directly from the eigenvalue problem for matrices AA^T, $A \in [A]$ (Deif, 1991). Given a central matrix $A^C \in R^{n\times m}$, find for interval matrix $[A] =\{A: |A–A^C| \le \Delta A\}$. (Here and in sequel, the absolute value $|\cdot|$ and the inequality sign '≤' are understood componentwise). A description of the set of singular values

$$\Sigma = \{s_1: A^TAx = s_1^2x,\ x\ne 0,\ A\in [A]\}.$$

In (Deif, 1991) the solution set has been received in analytic form. Let the matrix S_1^i is a diagonal matrix of the signs of the components of x^i, while S_2^i represents the signs of those of the matrix $2\ A^Cx^i +\delta Ax^i$, where $|\delta Ax^i| < 2|A^Cx^i|$. Then the squared singular values s_i^2 of $A^C + \delta A$,

$\forall |\delta A| \le \Delta A$, range the interval

$$[s_i^2((A^c)^TA^C – 2(S_1^i\Delta A^TS_2^iA^C)_{sym} + S_1^i\Delta A^T\Delta AS_1^i),\ s_i^2((A^c)^TA^C + 2(S_1^i\Delta A^TS_2^iA^C)_{sym} + S_1^i\Delta A^T\Delta AS_1^i)],$$

where B_{sym} denotes the symmetric part of a matrix B.

2.3 Adaptive Approach

Case 1. Let $[A] \in IR^{n\times m}$ – interval matrix. Then the singular values of $[A]$ are eigenvalues of the next matrix $[B]\in IR^{(n+m)\times(n+m)}$, $[B] = |B_1, B_2|$, where B_1, B_2 – are colums, $B_1 = |0, [A]|^T$, $B_2 = |[A]^T, 0|^T$.

Find a description of the eigenpair set

$$\Sigma = \{(\lambda, x)\in R^{n+m+1} | Bx = \lambda x,\ x \ne 0,\ B \in [B]\}.$$

Then we use the function (Mayer, 1992)

$$F\{(\Delta X, \Delta\Lambda)^T\} = –R\ \{([B]–\lambda E)x, Ix–1\}^T+\{E–RG\}\{(\Delta X, \Delta\Lambda)^T\},$$

where $\Delta X\in IR^{n+m}$, $\Delta\Lambda\in IR^1$, E, I-unit matrix and vector, $G =\{G_1, G_2\}$, $G_1 =\{([B]–\lambda E), I^T\}$, $G_2 =\{–\Delta X–x, 0\}^T$. If the enclosure holds

$$F\{(\Delta X, \Delta\Lambda)^T\}\subseteq (\Delta X, \Delta\Lambda)^T, \qquad (5)$$

then an eigenpair $[\lambda]$, $[x]$ of $[B]$ lies within

$$[\lambda] \in \lambda + \Delta\Lambda,\ [x] \in x + \Delta X.$$

In this case we calculate floating point approximations λ_i, $i = 1, \ldots, (n+m)$ for the eigenvaluees of $[B]\in IR^{(n+m)\times(n+m)}$, with the shifted QR algorithm. Consequently, we can iterate accoding to

$$(\Delta X, \Delta\Lambda)^{(k+1)} = F(\Delta X, \Delta\Lambda^{(k)}),\ k = 0,1,2,\ldots,$$

$$\Delta X^{(0)} =x,\ \Delta\Lambda =\lambda,$$

until (5) holds.

Case 2. $[A] \in IR^{n\times m}$ – interval matrix. Find a description of solution set of maximal singular value, left and right singular vectors for an interval matrix $[A]$:

$$\Sigma^P =\{(s_1, U, V) \in R^{m+n+1},\ s_1\in R^1,\ U\in R^m,\ V\in R^n:\ AU = sV,\ A^TV = sU,\ V^TV= 1,\ U^TU = 1,\ \forall A\in [A],\ A \text{ with property } P\}, \qquad (4)$$

where P is some fixed property such as symmetry, skew-symmetry, Toeplitz form, etc.. For example, let interval matrix $[A] \in IR^{m\times m}$ with $[a_{ij}] =[a_{ji}]$ for $i,j = 1,\ldots,m$. The set of matrices

$$\{A^{sym}\}=\{A\in R^{m\times m} | A\in [A],\ A \text{ symmetric}\}$$

is called a symmetric interval matrix. $\{A^{sym}\} \notin IR^{m\times m}$ is not interval matrix in the usual sense. $\{A^{sym}\} \subseteq [A]$ and $\{A^{sym}\} = [A]$ if and only if $a_{ij}^- = a_{ij}^+$ for $i,j = 1,\ldots, m, i \neq j$.

The aforementioned solution set is

$$\Sigma^{sym} = \{(s_1, U, V) \in R^{m+n+1}, s_1 \in R^1, U \in R^m, V \in R^n: AU = sV, A^TV = sU, V^TV = 1, U^TU = 1, A = A^T \in [A]\}. \quad (6)$$

Singular value decomposition (SVD) of interval matrix includes the following procedures:

Procedure 1. Definition of the set *(4)* of maximal singular values s_1 and corresponding left and right singular vectors for a real (a thin) matrix $\forall A \in [A]$.

For definition of maximal singular value s_1 and the left and right singular vectors U and V, corresponding to the s_1, the rather simple and reliable scheme and deflation method have been used (Tarakanov, 2000):

$$U^T_{(k+1)} = V^T_{(k)}A,\ V_{(k+1)} = AU_{(k+1)}$$
$$s_k = U_k^T A V_k, \quad |s_{k+1} - s_k| \leq \varepsilon, \quad (7)$$

where $k=0,1,2,\ldots$ - is the number of iteration, ε - the given precision of calculation. It can be shown, that for arbitrary unit vectors $V_{(0)}$, $U_{(0)}$ iterations by scheme *(7)* converge in general case to the singular vectors U, V corresponding to the maximal singular value $s_1 = V^TAU$. Calculate the interval hull for Σ^P *(4)*

$$\Sigma^P = \{[s_1], [U_1], [V_1]\}.$$

Then using *deflation method*, the next matrix

$$A_{(p)} = A_{(p-1)} - s_{p-1}U_{p-1}V_{p-1}^T \quad (8)$$

is formed on the step p, and its maximal singular value s_p and corresponding singular vectors U_p, V_p are determined by the scheme *(7)*. Calculate the interval hulles for Σ^P_p

$$\Sigma^P_p = \{[s_p], [Up], [Vp]\}.$$

Procedure 2. Definition of the intervall hulles for the sets of singular values and right and left singular vectors of interval matrix *[A]*. We define the matrix $B \in R^{m+n+2}$ for $\forall A \in [A]$ by (Alefeld, 1987):

$$B = \{B_1, B_2, B_3, B_4\}, \quad (9)$$

where B_i, $i = 1, 2, 3, 4$ the columns of corresponding dimensions

$$B_1 = |A, -sI_n, 2U^T, 0|^T,\ B_2 = |-sI_m, A^T, 0, 2V^T|^T,$$
$$B_3 = |-V, 0, 0, 0|^T, \quad B_4 = |0, -U, 0, 0|^T$$

and the vectors $v \in R^{m+n+2}$, $r \in R^{m+n+2}$ and $f(v) \in R^{m+n+2}$ by

$$v = |\Delta U, \Delta V, \Delta s, \Delta s|^T,\ r = |sV - AU, sU - A^TV, 1-U^TU, 1-V^TV|^T,$$
$$f(v) = |\Delta s \Delta U, \Delta s \Delta V, -\Delta U^T \Delta U, -\Delta V^T \Delta V|^T, \quad (10)$$

where $A \in [A]$, s, U, V –thereafter the maximal singular value and left and right singular vectors, the errors are equal $\Delta s = s - s'$, $\Delta U = U - U'$, $\Delta V = V - V'$, $\Delta s \in R^1$.

Let $v \in R^{m+n+2}$, $[v] \in IR^{m+n+2}$. As it is shown in (Alefeld, 1987) the matrix B is nonsingular. Assume that, L is an approximation to the inverse of B or the exact inverse of B itself. Consider

$$[F] = Lr + (E - LB)[v] + Lf([v]) \quad (11)$$

(E – unit matrix of corresponding dimention).

If the enclosure

$$[F'] \subseteq [v'], \text{ for } [v'] \subseteq IR^{m+n+2} \quad (12)$$

holds, then *([s],[U],[V])* of *[A]* lies within

$$[s] \in s + \Delta s', [U] \in U + \Delta U', [V] \in V + \Delta V'. \quad (13)$$

Consequently, we can iterate accoding to

$$(\Delta v^{(k+1)}) = F(\Delta v^{(k)}),\ k = 0,1,2,\ldots,\ \Delta v^{(0)} = v, \quad (14)$$

until *(12)* holds.

Now the following algorithm computes interval singular value *[s]*, left and right singular vectors *[U], [V]:*

- compute with iterations by scheme *(7)* maximal singular value s_1 and the corresponding left and right singular vectors U_1, V_1 for any $A \in [A]$;
- compute the interval hulle for Σ^P *(4);*
- calculate matrices *B, L;*
- iterate accoding to *(14);*
- calculate *(13).*

Obtained results of SVD of interval matrices were used for development of Pattern Recognition procedures (Tarakanov, 2000) for Interval Aftificial Immune Systems for Surveillance the Plague and Security Systems.

3 Interval AIS for Surveillance the Plague

Natural plague foci in the former soviet state Kazakhstan cover an area of 130 million hectares and over the past 50 years they have been considered the most active plague foci in the world. Recently, local and foreign workers have increased the level of human activity in the natural foci regions, often in connection with an intensive exploitation of natural resources (e.g. gas and oil). These activities are often organized by multinational companies, which increase the probability of plague cases being exported abroad.

Plague foci in Kazakhstan covering vast territories are characterized by different regulation mechanisms at the population species and community (biocenotic) levels. The plague epizootic process is a complex multicomponent dynamic system. The behavior of particular subsystems of the plague epizootic triad (agent-host-vector), the entire epizootic process in foci taking into consideration the complicated interrelations of the above subsystems have been investigated by microbiologists, biologists, epidemiologists, etc (Ageyev, 1975, Aikimbayev, 1994, Aubakirov, 1990,…) and consequently, considerable experimental material has been accumulated on population, organism and cellular levels.

For the first time, mathematical models have now been obtained (Marshall, et al., 2001) which show the dynamics of interactive stray-host relationship on

population level in the plague triad. This solved the problem of structure and parametric identification in a group of non-linear stochastic differential equations taking into account the delay and influence of external factors as an additive disturbance with defined statistical characteristics. The results of simulation modelling were obtained and the results of digital experiments were compared with real data.

A new qualitative approach to the solution of the task of prediction of epizootic processes in natural plague foci was suggested (Tarakanov, Sokolova et al., 2000), using the mathematical basis of AIS for its solution. As was mentioned above, it was based on singular value decomposition (SVD) in combination with the deflation method, the binding energy between a pair of formal peptides. The characteristics of SVD are its quick convergence and robustness, which allows a considerable quantity of spatially-temporal series (45 temporal series) characterizing the condition of triad on the different levels for the solving of the prediction task. Using procedures for supervised learning, unsupervised learning, classification and presentation of the results of research in IC shape space. The comparative analysis of this approach compared with traditional approaches has demonstrated its considerable advantages. The results from the creation of an Interval Artificial Immune System for the surveillance of plague are shown below.

3.1 Significant Factors and the Characteristics of Plague Processes

The plague epizootic process is a complex multicomponent dynamic system which is characterized by the interaction of particular subsystems of the plague epizootic triad: agent-host-vector (Aikimbayev et al., 1994, Aubakirov et al., 1990, Ageyev, 1975, Marshall et al., 2001).

The state of the agent is characterized by the following differential and diagnostic properties (Aikimbayev et al., 1994). The Qualitative Properties: the morphology of colonies the susceptibility to bacteriaphage (Pokrovskaya's homologous, heterologous pseudotuberculous), glycerin, fermentation, rhamnose fermentation, denitrification/nitrification, pesticinogenecity, the susceptibility to pesticine, the presence of VW-antigens, the need of growth factors. The Quantitative Properties: the dependence on calcium at 37^0C, the presence of antigen of Fraction 1 in the reaction of passive hemagglutination and immunoglobulin plague erytrocytic diagnosticum, growth on the medium with hemin, the integral property - virulence in white mice and guinea-pigs (LD-50, DCL, according to Kerber's calculation). Most frequently, the state of the agent can be characterized only by its numbers expressed through indirect indices: the infestation of rodents, fleas or samples (points) obtained not only from the given area but also from the adjacent ones.

The state of fleas - vectors of a plague microbe - is expressed (Ageyev, 1975) through their numbers, the seasonal activity in attacking animals, through the sex and age composition of the imaginal phase; as well as with the help of mass emergence of imago on animals and in openings of rodent burrows after hibernation, the ovipositor (its beginning, peak and termination), the larvae hatch (its beginning, mass numbers and termination) and other indices. Other relative indices of numerocity include the Index of abundance, the Index of dominance, and the Index of fidelity, the total numbers of fleas per hectare.

The state of hosts is characterized by the aggregate of factors of influence, the set of characteristics at the biocenotic, population and organism levels. This set is included the following main factors: density of family burrows, numbers of the hosts in different seasons, age and sex structure of host population; reproduction, mortality rate, dynamics of the level of infection susceptibility, determination of age, sex and generative state of hosts, etc.

The data characterizing the state of the members of the epizootic triad are classified according to the season (spring, summer, autumn), and sometimes (for example, in case of reproductive indices) according to months and even decades.
Weather, geographical and space characteristics constitute the indices of external factors**.** The weather characteristics are: the temperature of the air, soil and precipitation, various hydrothermal coefficients, the recurrence and the wind velocity according to compass points, the recurrence of the types of atmospheric circulation, the value of flood, etc.

3.2 The tasks of Pattern Recognition

The Development of AIS is important to recognise future patterns of plague epizooyic process (Tarakanov, Sokolova et al., 2000). AIS is intended to improve risk analysis and underlying understanding of the space-time dynamics of the plague in the Republic of Kazakhstan. Moreover, AIS could be considered as a part of the surveillance systems of re-emerging infectious diseases of direct importance to the world community both through tourism and other international activities. AIS could be considered as a model system for processing surveillance data of other dangerous infections in all parts of the world.

Input data for the AIS are generated by the two main blocks:

1) Base of surveillance data on the plague on Akdala plain, including computerizing the existing historic data, which is currently only available on handwritten paper;
2) A set of mathematical models (stochastic, interval and discrete) of the space-time dynamics of the plague.

Based on the input data the main functions of the AIS consist in evaluating a current danger of the plague

infection, as well as predicting a risk of infection in the future. To perform such functions the core of the AIS represents a pattern recognition block. Namely software of this block emulate our IC approach to pattern recognition.

3.2.1 Supervised Learning

To identify the state of the epizootic process we use the input data to represent the state of host and vector, the indices of external factors andthe space-time dynamics of the plague.

Figure 1 and Table1 give input data for this task. Figure 1 shows numbers of infected sectors (vertical axe) over the several years (horizontal axe) for the Akdala plain. Each sector represents a square of 10×10 km. If the plague's host (rodent), which has been caught in the sector, contains a plague microbe, then the entire sector considered as infected.

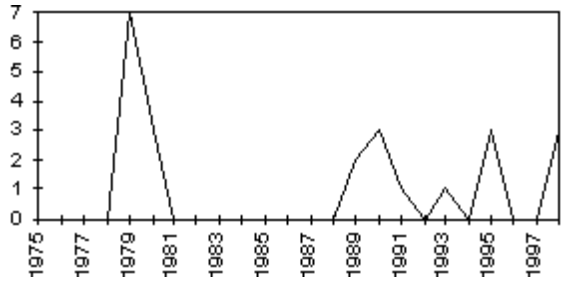


Figure 1: Space-time dynamics of the plague on the Akdala plain

It worth to denote, that the number of infected sectors, obtained by such a method, represents nowadays the most appropriate indicator of epizootic process (Aikimbayev et al., 1994).

A fragment of the surveillance data of the Akdala plague focus during 11 years is represented in Table 1.

Table 1: A fragment of the database of surveillance the plague

YEAR	**1976**	**1977**	**1978**	**.**	**1987**	**1988**
p1	[2.5, 3.3]	[1.3, 1.8]	[5.2, 6.1]		[4.0, 5.3]	[13.5, 15.8]
p2	[0.3, 0.7]	[4.1, 5.4]	[29, -32]		[4.4, 4.9]	[16.1, 18.6]
p3	[1.9, 2.3]	[1.1, 1.4]	[3.4, 3.9]		[1.7, 1.8]	[4.8, 5.0]
:	:	:	:	:	:	:
p40	33.9	12.0	34.2		36.2	15.3
p41	1.3	10.0	11.7		0.7	7.7
p42	5.0	6.0	8.7		1.0	25.7
p43	0.3	0.0	0.0		1.3	12.0
p44	0.0	0.0	0.7		1.0	1.0
p45	5.0	4.0	2.7		2.3	1.0

The fragment includes 45 parameters p1-p45 such as: number of rodents per square p1 (in autumn), p2 (in spring), number of infected rodents per square p3 (in autumn), p4 (in spring),…, total atmospheric precipitates– p40(September), average height of the snow blanket - p41 (January) – 43(Mach), p44 (November) – p45(December). As is shown in Table 1 such parameters as the number of rodents and fleas per square, number of infected rodents and fleas per square are interval. For example, the number of rodent per square in autumn – p1 equal *[2.5, 3.3]* in 1996, *[13.5, 15.8]* in *1998* and so on, (Tarakanov, Sokolova et al., 2000) are necessary for Pattern Recognition for interval systems.

For decision the task of the identification of epizootic process state we use four classes (Figure 1). Class 1 - from 1975 to 1978 – the period of depression prior to the epizootic process; Class 2 – from 1978 to 1979 –the ascending branch of the epizootic process; Class 3 from 1979 to 1981 – the descending branch of the epizootic process; Class 4 – from 1981 to 1985 the depression after the epizootic process. The states of the epizootic triad on the ascending branch and the descending branch of the epizootic processes are different. Thus, learning patterns are given by the years 1976-1981 and the task is to assign the corresponding classes to the years 1984-1998. Classes that have been assigned by the AIS to the years 1976-1981 are considered as a test of the process.

When the beginning of epizootic process has been predicted, the identification of the “power” epizootic process has been accomplished. We use Figure 1 and three classes: Class 1 – 1979, Class 2 – 1990, Class 3 – 1993. Thus, learning patterns are given by the years 1979, 1990, 1993 and the task is to assign the corresponding classes to years 1994 –1998. In this case 19 space-time

rows characterized the state of agent are added in Table 1(p 46 – p 64).

Using Immunocomputing allowed us to solve the tasks of host (large gerbil) and vector (fleas) number prognostics. For this purpose we use the annual (spring, autumn) space-time dynamics of the plague numbers representatives on Figure 2.

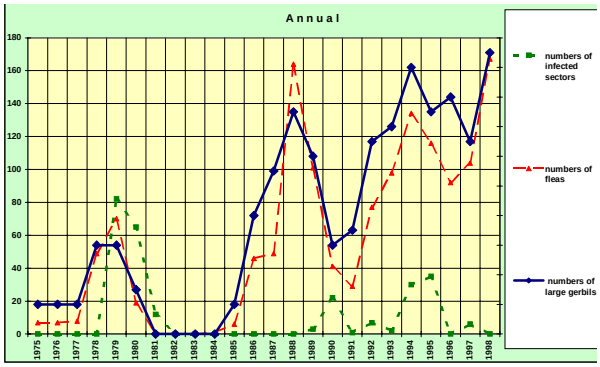


Figure 2: Annual space-time dynamics of the plague numbers

According to the immunocomputing approach the aforementioned tasks are solved as follows.

Foldering vectors to matrices

Fold every column R_{year} of Table 1 (vector of the dimension 45×1) to a matrix A_{year} of the dimension 9×5. As is obvious from Table 1 any elements R_{year} are interval elements so the matrix A_{year} also is interval.

Learning

Form matrices $[A_1]$, $[A_2]$, $[A_3]$,, $[A_4]$ for the classes $1,2,3,4$ correspondingly:

$[A_1] = \{[A_{1975}], [A_{1976}], [A_{1977}], [A_{1978}]\};$

$[A_2] = \{[A_{1978}], [A_{1979}]\};$

$[A_3] = \{[A_{1979}], [A_{1980}], [A_{1980}], [A_{1981}]\};$

$[A_4] = \{[A_{1981}], [A_{1982}], [A_{1983}], [A_{1984}], [A_{1985}]\},$

where $\{[S]\} = \{infS, sup\ S\}$

and midpoint $mid[A_i]$ of interval matrices $[A_i]$, $i = 1,2,3,4$.

Using the computing prosedures of Part 2 we compute their interval singular vectors:

$\{U_1,V_1\}$ – for $[A_1]$, $\{U_2,V_2\}$ – for $[A_2]$

$\{U_3,V_3\}$ - for $[A_3]$, $\{U_4,V_4\}$ - for $[A_4]$.

$$A_1 = \frac{A_{1977} + A_{1978}}{2},$$

Recognition $A_2 = A_{1979}$,

$$A_4 = \frac{A_{1984} + A_{1985}}{2}$$

Compute four values of the binding energy for every input pattern $[A_{year}]$ (upper case ^{T} designates a symbol of the transposing):

$$[\omega_1] = U^T{}_1[A_{year}]V_1, \qquad [\omega_2] = U^T{}_2[A_{year}]V_2,$$

$$[\omega_3] = U^T{}_3[A_{year}]V_3, \qquad [\omega_4] = U^T{}_4[A_{year}]V_4.$$

Determine the means of the minimum binding energy for every class:

$$\omega = min_{\omega\in[\omega]} max\ \omega_{A\in[A]}$$

Determine the class to be found by the minimal value of the binding energy:

$$k{:}\omega_k = min\ \{\omega_1,\omega_2,\omega_3,\omega\}.$$

The results of the recognition are showed in Table 2, where the minimal values of the energy are underlined. According to Figure 2, the classes that have been recognized could be treated also in the last column of Table 2 as a risk level of the plague infection.

Table 2: Results of the recognition

YEAR	CLASS by EXPERTS	- ω_1	- ω_2	- ω_3	CLASS by IAIS	RISK of INFECTION
1976	1	138	13	126	1	mid
1977	1	133-	18	113	1	mid
:	:	:	:	:	:	:
1985		141	10	148	3	low
1986		152-	66-	150-	1	mid
1987		174-	182	171	2	high
1988		197-	493-	183	2	high

3.2.2 Unsupervised Learning

Consider the interval matrix $[A]= \{[X_1],... ,[X_m]\}^T$ of dimension $m\times n$ formed by m interval vectors $[X_1]$, ..., $[X_m]$. Using the results of Part 2 calculate the SVD of this interval matrix:

$$[A] \subseteq [s_1][U_1][V_1]^T + [s_2][U_2][V_2]^T + ..., \qquad (15)$$

where $[s_1]$, $[s_2]$ are the first two interval singular values, and $[V_1]$, $[V_2]$ are the interval right singular vectors.

According to (Tarakanov, 1999) there can be established a rigorous correspondence between vector and Formal Peptide (FP). So, consider two FPs as antibodies: {FP-1, FP-2}, which are corresponded to vectors $[V_1]$, $[V_2]$. Consider also eleven FPs: $\{FP_1 , ... , FP_{11}\}$, which are corresponded to the strings of the matrix A (columns of Table1). Then every string A_i, which represents the number of year $i = 1,...,11$, can be mapped to the 2 values $\{x1_i , x2_i\}$ of the binding energy between FP_i and two antibodies:

$[x1_i] = [\omega(\text{FP-1}, FP_i)]$, $x1_i = min\{x1_i{:}\ A_i \in [A_i]\}$

$[x2_i] = [\omega(\text{FP-2}, \text{FP}_i),. \quad x2_i = min\{x2_i,: A_i \in [A_i]\}$

The obtained results are given in Table 3.

Table 3: Classification of the years by a shape space of the IAIS

YEAR	*x1*	*x2*	CLASS	RISK of INFECTION
1976	0.126	0.359	C2	
1977	0.113	0.337	C3	
1978	0.194	0.313	C1	
1979	0.5820	-0.261	A	High
:	:	:	:	:
1986	0.197	0.292	C1	
1987	0.348	0.188	B	Mid
1988	0.699	-0.366	A	High

The results presented in this Section show, that Immunocomputing is rather powerful, robust and flexible approach to complex biological systems with interval data such as Natural Plague foci.

4 Interval AIS for Security System

The protection of the person and protection of property is always one of the main problems facing a society during its historical development as these questions are most closely connected to one of the basic instincts of the person - an instinct of self-preservation.

Responsibility for the protection and also for the prevention of criminal activity is the responsibility of the state through law enforcement bodies and the justice system.

The protection of official buildings, industrial enterprises, banks, places of retail trade, homes etc. often utilizes security systems. These systems have the all-characteristic indications of complex systems:

- large number of the interconnected elements,
- basic indeterminacy from of indefinity of the information about a potential criminal and his operations;
- "human factor" connected with the necessity of acceptance by the man of operating solutions;
- natural factors consisting in wide variety of climatic conditions,
- natural and industrial parasites.

The analysis of domestic experience and foreign radiants (C and K Systems, 1997, etc.) has shown, that approximately 95 % of alarms are false. Many malfunctions are caused by a device reacting to various false indicators, which should be referred to as parasites. The analysis of complex security systems has shown what it is often impossible to determine the reasons for a malfunctioning of the signal system.

The use of the IC approach for an intelligent technique of analysis of non-standard alarm information has been developed and an Interval AIS for Security System (ISS) for the complex objects has been constructed (Sokolova et al., 2000). It has considerably lowered the number of malfunctions of signal security systems.

ISS requires the realization of the following functions:

- state estimation of the complex objects and analysis of the non-standard alarm information on the basis of the mechanism of information processing;
- implementation of procedures of supervised and unsupervised learning;
- implementation of pattern recognition procedures of the non-standard alarm information in the shape space of IC.

Passive infra-red gauges (PIR-detectors) have received a wide circulation and are one of the basic means of the signal system for protection of premises, areas, passes, corridors etc. A passive principle of action, i.e. the absence of any radiation, makes this method absolutely harmless to the person who is in the zone of its action. It is also used for the protection of exhibits, for example, in a museum exhibition of ancient manuscripts, fabrics and other fragile and easily destroyed objects. The principle of action of a PIR-detector is based on measurement of the difference in temperatures of the person and the surrounding environmental surface (walls, a floor and furniture) premises witrhin the visibility range of the optics of the device.

The main failing of PIR-detectors is the difficulty of identifying the point of penetration, and the size of an intruder. Identifying the point of penetration is necessary for an effective reaction by security staff. The correct estimation of the size of the intruder can sharply reduce the number of false alarms. The cause of an alarm can be various animal (cat and dog), left by the owners in a location. The mode of creation of the open information channel for each feeler of the PIR-detector with matrices of sensitive elements 4×4, 8×8, 16×16 is applied for elimination of this shortfall ininformation. Such devices permit identification of the point of penetration of the intruder and sharply reduce the number of false alarms from the signal system. Depending on the means used for detecting intrusions, the corresponding factors may accept the values appropriate logic 0 or 1 (threshold algorithms) or to change in any fixed interval (for example, in a range from 0 up to 255 at application of 8-bit analog-digital converters).

Using PIR-detectors with a through information channel it it is possible to estimate the route an intruder is taking into a protected zone. Accordingly, having such information, will enable security staff to define tactics to

identify his objective and prevent the escape of the intruder.

Let's consider that identification of the point of penetration in a guarded zone and the possible route to of an intruder to his objective are carried out with the help of PIR-detector with a matrix 4×4 and a through information channel. Let's assume, that in the fixed instants with application space mechanical, or optical scanning the alarming information from the PIR-detector with a matrix 4×4 is obtained, i.e. the vectors of values of indications of dimension 16×1 are obtained thereafter in times t_1, t_2, t_3:

$$X_1 = [1,0.1,0.4,1,1,1,0.2,1,0.1,1,1,0.7,0.3,0.8,1,0.2]^T,$$

$$X_2 = [0.1,0.6,0.2,1,1,1,0.4,1,0.1,1,1,0.7,1,0.8,1,1]^T,$$

$$X_3 = [1,0.1,0.8,1,1,0.5,1,1,0.1,1,1,0.4,0.2,0.4,1,0.5]^T.$$

On the basis of the obtained information and in view of judgement of the expert the learning sampling of alarming situations as the Table 4 is formed.

Table 4: Learning sampling by results of the non-standard alarming situations analysis.

NUMBER	VALUES OF INDICATIONS					CLASS
	x_1	X_2	x_3	..	x_{16}	
1	1	0,3	1		1	1
2	0,1	0,6	1		1	2
3	0,2	0,8	0,9		0,7	3
4	0,5	1	0,7		0,1	4
N	1	1	0,5		0,1	1

From these received vectors we form the interval vector $[X]$ as

$$[X] = \{X_1, X_2, X_3\}.$$

Then we use all steps of Supervised Learning, Unsupervised Learning.

Applying interval AIS to Security Systems has allowed the use of dynamic information about non-standard alarm situations and considerably reduced the quantity of false signals (Sokolova, ed. A.O. Tarakanov, 2000).

5 Conclusions

The results presented in this paper show, that Immunocomputing is a powerful, robust and flexible approach to complex systems with interval uncertainty. Further development of the Immunocomputing approach on class of interval objects requires the development of additional simplified computer procedures of singular decomposition of interval matrices, production of inner and outer estimation of a set of solutions and the values of interval bilinear forms that determine the binding energy between formal proteins.

5.1.1 Acknowledgments

The authors acknowledge the support of the EOARD under the project 2200 p “Development of Mathematical Models of Immune Networks Intended for Information Security Assurance”, EU Commission under INCO-contract № ICA2-CT-2000-10048 “The plague of Central-Asia an epidemiological study focusing on space-time dynamics”.

5.1.2 References

V.S. Ageyev (1975). *Parasitic contacts of rodents in the river valleys of the desert zone of Kazakhstan and their significance in the plague epizootology*. Saratov, 3-19 pp. (in Russian).

A.M. Aikimbayev, et al. (1994). *Epidemiological Plague surveillance in the Ural-Emba and Ustyurt Autonomous Foci.* Almaty, Gylym, (in Russian).

G. Alefeld, and J. Herzberger (1983). *Introduction to Interval Computation.* New York, Academic Press.

G. Alefeld (1987). Rigorous Error Bounds for Singular Values of a Matrix Using the Precise Scalar Product. In E. Kaucher, U. Kulish and Ch.Ullrich, (eds.), *Computerarithmetic:* 9-30. Teubner, and Stuttgart.

S.A. Aubakirov et al. (1990). *Instruction on Landscape-Epizootic Regionalization of Natural Plague Foci in Central Asia and Kazakhstan.* Alma-Ata, Gylym, (in Russian).

D. Dasgupta, (ed.) (1999). *Artificial immune systems and their applications*. Springer-Verlag, Berlin, New York.

G. Mayer (1992). Enclosures for eigenvalues and eigenvectors. In L. Atanassova and J. Herzberger (eds.), *Computer Arithmetic and Enclosure Methods*, North-Holland, Elsevier Science Publishers B. V.

E.C. Marshall, A. Frigessi, N.C. Stenseth, M. Holden, V.S. Ageyev and N.L. Klassovskiy (2001). *Plague in Kazakhstan: a Bayesian model for the temporal dynamics of a vector-transmitted infectious disease.* Oslo, University of Oslo.

S.P. Shary (1995). Solving the linear interval tolerance problem. *Mathematics and Computers in Simulations*, **39**, 53 – 85.

S.P. Sokolova et al., (ed. A.O. Tarakanov) (2000). *Intelligent Security Systems*. Almaty, Police Academy of Kazakhstan (in Russian).

A.O. Tarakanov (2001). Information Security with Formal Immune Networks. *Proc. of the Methods, Models and Architectures for Network Security (MMM-ACNS 2001),* 115-126, St. Petersburg.

A.O. Tarakanov, S.P. Sokolova, B.A. Abramov and A.M. Aikimbayev (2000). Immunocomputing of the natural plague foci. *Proc. of the 2000 Genetic and Evolutionary Computation Conference (GECCO-2000)*, 38-41, Las Vegas, USA.

Hierarchy and Convergence of Immune Networks: Basic Ideas and Preliminary Results

Leandro N. de Castro
Department of Computer and Electrical Engineering
Faculty of Computer and Electrical Engineering
State University of Campinas, Brazil.
lnunes@dca.fee.unciamp.br

Jon Timmis
Computing Laboratory
University of Kent at Canterbury, UK.
J.Timmis@ukc.ac.uk

Abstract

aiNet is an artificial immune network model originally developed to perform automatic data compression. Combined with graph theoretical and statistical clustering techniques, aiNet is a powerful data clustering and classification tool. However, the original aiNet model suffers from the lack of a well-defined stopping criterion and an ad hoc approach to parameter initialization, prior to the training process. This paper has two main goals. First, by assessing convergence criteria employed in a class of artificial neural networks, a suitable stopping criterion can be created for aiNet. Secondly, the paper demonstrates that through the use of a cooling schedule for some of these user-defined parameters, it is not only possible to reduce the importance of their initial values, but also this leads to possible derivation of a hierarchical tree of immune networks. Due to the very limited space available, only the basic ideas of a novel convergence criterion, and an approach to develop a tree of aiNets will be presented, together with an illustrative example.

1 INTRODUCTION

The emerging field of artificial immune systems (AIS) has grown very rapidly in the last few years. The applications of AIS can be considered as very diverse, ranging from autonomous navigation to data analysis (Dasgupta, 1999; de Castro & Timmis, 2002). Two immune network models developed quite independently by Timmis (2000) and de Castro & Von Zuben (2000) have been used as alternative biologically motivated approaches with which to perform data clustering.

Other, more well established biologically motivated paradigms, widely used to perform data clustering are the self-organizing neural networks, in particular the self-organizing feature maps (SOFM) introduced by Kohonen (1982). The SOFM, as originally proposed, is capable of performing a dimensionality reduction of a set of input data into a (usually) regular grid of output units. During the SOFM learning process, 'similar' input data are mapped into neighboring units in the output grid of the network. By adopting a given metric to evaluate the similarity among the input data, combined with the network weight vectors (usually the Euclidean distance), at the end of the learning process the output grid of the SOFM is capable of preserving topological and metric relationships contained within the input data set. Despite great potential, the original SOFM suffers from some limitations. First, although the SOFM is capable of mapping similar data items into neighboring units in the output grid, the automatic inference of the number of clusters contained in the data set is not a straightforward process. Second, the determination of a suitable dimension for the output grid is also not automatic. Finally, the mapping performed by the SOFM does not account for any hierarchical structure within the data set.

In order to alleviate some of these limitations of the original SOFM, several variations have been proposed and introduced in the standard algorithm. Among these are methods for the automatic segmentation of clusters within a trained SOFM via, for example, the U-matrix (Ultsch, 1995); algorithms to dynamically generate the network architecture according to the input data set (Fritzke, 1994; Cho, 1997; de Castro & Von Zuben, 1999); and the proposal of hierarchical methods for growing neural network trees (Adams et al., 1999; Costa & Netto, 1999).

The immune network model discussed in this paper was introduced by de Castro and Von Zuben (2000), and named aiNet (Artificial Immune NETwork). The main role of the standard adaptive algorithm proposed for the aiNet was to reduce data redundancy, whilst at the same time extracting relevant information from the data set, such as the spatial distribution of the inherent data clusters. The network cells within aiNet are represented in a space of same dimension as the input data, i.e. no dimensionality reduction is performed, but the network size is controlled based upon the immune network dynamic and metadynamic processes (Varela et al., 1988). The network cells represent an "internal image" of the input data set, and therefore it became necessary to use additional tools in order to automatically identify and separate clusters in this network of cells. The authors employed the use of the minimal spanning tree (MST), borrowed from graph theory, as a useful mechanism with which to automatically detect and separate the network clusters (de Castro & Von

Zuben, 2001). In order to assess the performance of aiNet, the authors applied the aiNet model to the well-known two spirals problem (Fahlman & Lebiere, 1990) and also to the chain link problem (Ultsch, 1995). Although these datasets are composed of non-linearly separable clusters of data (which cannot be automatically detected with the standard SOFM), the inner-subset distance of a data point is of orders of magnitude smaller than the inter-subset distance. This makes the MST application very effective at processing networks produced from aiNet, as aiNet positions network cells in appropriate locations within the space. Additionally, the number of clusters was low (only two, in both cases) and their shapes very similar. In cases where the number of clusters is high and their shapes are non-uniform, the proposed MST approach for the aiNet model still presents good results, but may not be capable of solving the whole problem in a single run for a given set of user-defined adaptive parameters.

This paper proposes two new theoretical results for the aiNet algorithm. First, it proposes a convergence criterion for the network iterative process that seeks to interrupt learning when the capability of the network to represent the input data degrades for a given network dimension. Secondly, the paper introduces an automatic hierarchical method to generate a tree of aiNets capable of detecting clusters with less-uniform characteristics, alleviating the problem of choosing initial values for some user-defined parameters. This model was largely inspired by the hierarchical variations proposed for the SOFM and referenced above, in particular those by Adams et al. (1999) and by Costa and Netto (1999).

This paper is organized as follows. Section 2 briefly discusses the problem of cluster analysis and competitive learning. This is important to allow for a characterization of aiNet and to provide a background for the (convergence and hierarchical) methods to be proposed. Section 3 briefly reviews the aiNet learning algorithm. Section 4 proposes a convergence criterion for aiNet and Section 5 proposes the hierarchical approach for the generation of a tree of aiNets. Section 6 illustrates the performance of the methods proposed when applied to a simple benchmark problem. The work is concluded in Section 7 with a discussion of the methods proposed and possible extensions of this work and future trends for the aiNet algorithm.

2 CLUSTER ANALYSIS AND COMPETITIVE LEARNING

This work addresses the problem of detecting inherent separations among subsets (clusters) of a given data set, named generically **Ag**, in a shape-space governed by an Euclidean distance. Note that inherent separation is used here to emphasize that any separation detected by the aiNet minimal spanning tree (Section 3.1) is going to depend solely on interpoint distances within the resultant network cells.

The partition of data into subsets, which are less heterogeneous than the primary set, has applications in several domains, where the main goal is to get some insight of the underlying structure of the data. Although several different algorithms have already been proposed in the literature, no general framework has yet been presented. For good surveys the interested reader might refer to Everitt (1993) and Jain et al. (1999).

Clustering methods range from largely heuristic approaches to more formal procedures, and are commonly divided into hierarchical and partitioning methods. The discussion to be presented here focuses on the hierarchical methods, whose limitations were sources of inspiration for the improvements proposed in this paper. Traditional optimization clustering techniques, such as k-means clustering, discover the clusters in the data by optimizing the initial cluster prototypes to reduce a cost function. The most commonly used cost function is the mean squared error (MSE), which is given by

$$\text{MSE} = \Sigma_{\mathbf{Ag}} \|\mathbf{Ag} - \mathbf{Ab}_{i*}\|^2, \quad (1)$$

where **Ag** is an input vector, and $\mathbf{Ab}_{i*}$ is the classifying prototype such that

$$\|\mathbf{Ag} - \mathbf{Ab}_{i*}\| \leq \|\mathbf{Ag} - \mathbf{Ab}_i\| \quad \forall i, \quad (2)$$

where $\mathbf{Ab}_i$ are the prototypes, which in this case correspond to the aiNet cells.

A major drawback of these clustering techniques is that they require the pre-definition of the number of clusters to be used, and are also sensitive to the initialization of the prototypes.

Hierarchical clustering techniques provide a complete, structured grouping of the input data set, going from all objects being members of the same cluster, to several clusters each containing a single datum object. The hierarchical methods can be further subdivided into agglomerative and divisive strategies. Agglomerative methods start with all objects having their own cluster and combine clusters until a single cluster exists. In contrast, divisive methods begin with all objects being members of the same cluster and divide the clusters until every cluster has just one object. The main problems with the hierarchical methods are (1) undesired merging of objects cannot be corrected at later stages; (2) memory usage required is usually proportional to the square of the number of clusters in the initial partition; and (3) the results may be hard to interpret, particularly the case for large data sets.

Finally, in aiNet the input data are assumed unlabeled, thus resulting in a type of competitive learning algorithm. With competitive learning, clustering of the input data is performed such that the MSE given by Eq. (1) is iteratively reduced. For each input datum (**Ag**), a winning unit (the closest to the input, given a certain distance measure) is found ($\mathbf{Ab}_i$), and is moved towards the input vector using the following updating rule:

$$\mathbf{Ab}_i = \mathbf{Ab}_i + \gamma(\mathbf{Ag} - \mathbf{Ab}_i), \quad (3)$$

where γ is a learning rate.

3 aiNet DESCRIPTION

One of the most striking characteristics of the immune system is its capability to recognize and eliminate harmful invading micro-organisms (e.g., viruses, bacteria, and parasites) or malfunctioning self cells (e.g., tumor and cancer cells). Clonal selection and expansion is the most accepted theory used to explain how the immune system copes with these invading micro-organisms, broadly named antigens. In brief, the clonal selection and expansion theory states that when antigens invade an organism, a subset of the immune cells capable of recognizing these antigens proliferate and differentiate into active or memory cells (Burnet, 1959). The active cells have the primary role of combating the invasion, while the memory cells have long life spans. An interesting phenomenon that occurs during the cellular proliferation is a mutational event with high rates. This mutation process, together with a strong selective force, ensures that the set of memory cells has improved capabilities of recognizing the antigens. As the total number of immune cells contained in an organism is limited and the number of possible invaders is almost limitless, the immune system has to be capable of generating enough cellular diversity. In addition, it has to be capable of extracting some general information (common patterns) contained in these invading antigens so as to promote more effective responses in cases of future expositions. This information extraction process is a consequence of the mutation, selection and maintenance of memory cells.

In contrast to clonal selection, the immune network theory is often explored to describe how some components of the immune system (cells and molecules) interact with one another even in the absence of external stimuli. The immune network hypothesizes that the immune system presents an intrinsic eigen-behavior. This suggests that self-recognizing processes can 'suppress' the network dynamics (activity), while the recognition of foreign micro-organisms leads to cell proliferation and network activation (Jerne, 1974).

These two immune theories are the basis for the artificial immune network (aiNet) model proposed by de Castro & Von Zuben (2000, 2001). In aiNet, the recognition of an input pattern (antigen) results in cell proliferation, mutation and selection as suggested by the clonal selection theory. The recognition of components of the network itself results in the network suppression; a process simulated by the elimination of all but one of the self-recognizing cells. By following these two immune principles, the aiNet is capable of extracting relevant features contained in a set of input data at the same time it eliminates data redundancy.

Using a functional and high-level description, the aiNet learning algorithm can be presented as follows. For a detailed algorithm, the interested reader is invited to refer to de Castro & Von Zuben (2001):

1. *Initialization*: create an initial random population of network antibodies;
2. *Antigenic presentation*: for each antigenic pattern, do:
   - 2.1 *Clonal selection and expansion*: for each network element, determine its affinity with the antigen presented. Select a number of high affinity elements and reproduce (clone) them proportionally to their affinity;
   - 2.2 *Affinity maturation*: mutate each clone inversely proportional to affinity. Re-select a number of highest affinity clones and place them into a clonal memory set;
   - 2.3 *Clonal interactions*: determine the network interactions (affinity) among all the elements of the clonal memory set;
   - 2.4 *Clonal suppression*: eliminate those memory clones whose affinity is less than a pre-specified threshold;
   - 2.5 *Metadynamics*: eliminate all memory clones whose affinity with the antigen is less than a predefined threshold;
   - 2.6 *Network construction*: incorporate the remaining clones of the clonal memory with all network antibodies, resulting in a matrix **M** of memory antibodies;
3. *Network interactions*: determine the distance between each pair of network antibodies and store these data in a matrix **D**;
4. *Network suppression*: eliminate all network antibodies whose affinity is less than a pre-specified threshold;
5. *Diversity*: introduce a number of randomly generated cells into the network

Cycle: repeat Steps 2 to 5 until a pre-specified number of iterations is performed.

3.1 EXTRACTING THE CLUSTERS

The aiNet learning algorithm described above was designed to generate a reduced set of cells (according to the suppression threshold – σ_s) representative of the spatial distribution of the input data. After defining the set **M** of cells' coordinates, along with their corresponding distance matrix **D**, it is necessary to interpret the resultant network given by the 2-tuple $\langle \mathbf{M}, \mathbf{D} \rangle$. This includes the determination of the number of clusters contained in the resultant network (indirectly in the data set) and the selection of which network cells compose each of these detected clusters.

The authors employed the minimal spanning tree (MST) method as proposed by Zahn (1971) to detect clusters in a trained aiNet. Basically, this method traces the MST among the resultant cells in the network and searches for inconsistent edges in the MST. In this case, inconsistency refers to the *ratio r*, named *inconsistency ratio*, between the length of a given edge and a number of neighbor edges on both sides of the selected edge. The method suggests that an edge that is much larger than the average of nearby edges gives an indication of the existence of more than one cluster in the data.

4 CONVERGENCE PROPERTIES

Motivated by the desire to introduce more rigorous stopping criteria for immune networks, it was necessary to study the convergence properties of aiNet. It was first necessary to characterize which type of adaptation it performs, and also to study techniques from cross-validation theory, which is sometimes employed in supervised learning for neural networks.

As previously discussed , the aiNet learning algorithm has two distinct, but interrelated, processing stages. In the first stage, aiNet behaves in a clonal selection fashion, and in the second stage, in a network-based form. The clonal selection part follows the clonal selection algorithm, named CLONALG, proposed by de Castro and Von Zuben (2002). This algorithm can be considered as an evolutionary-like algorithm as it is a population-based search technique, where the individuals of the population reproduce, their offspring suffer genetic variation and are then subjected to selection. An interesting aspect of this algorithm is that mutation is a guided process, aiming at increasing the capability of recognition of the network cells, in relation to the input data (antigens). The guided mutation strategy follows the same competitive rule as given by Eq. (3), and is found in self-organizing neural networks, like the SOFM. The main difference with aiNet is that the learning rate has a distinct value for each cell in the network, and this rate is proportional to the distance among the selected aiNet cells (and their respective clones) and the current input datum (antigen). Thus, the aiNet learning algorithm can be characterized as an evolutionary algorithm with self-organizing learning.

This type of dynamics within the network, makes it very difficult to formally prove the convergence properties of aiNet. In the case of the SOFM, there are several works in the literature regarding its equilibrium states and convergence properties (Kohonen, 1982; Ritter & Schulten, 1988; Erwin et al., 1992). Although both networks, the SOFM and aiNet, are self-organizing, several differences between them can be highlighted. The SOFM assumes a pre-defined neighborhood function among neurons, it performs a dimensionality reduction of the input data and it employs a cooling schedule for some parameters, such as the learning rate and neighborhood radius. The use of these cooling schedules for the learning rate and neighborhood radius are fundamental for guaranteeing that the SOFM will achieve a stationary convergent state. The aiNet in contrast, performs a sort of greedy search for an 'optimal learning rate' for each selected network cell and its clones, rather than iteratively cooling the learning rate. Additionally, instead of an explicit neighborhood function, the aiNet selects a set of k-nearest neighbors (KNN) to a given input datum; these KNN are taken from the first to the last iteration of the algorithm and is undertaken in the affinity maturation stage of the algorithm. Additionally, there is no cooling schedule applied to this number of nearest neighbors.

From the viewpoint of an evolutionary-like behavior for aiNet, convergence proof becomes an even more difficult task. Evolutionary algorithms incorporate complex nonlinear stochastic processes, such that the theoretical convergence analysis can only be performed using methods like Markov chains, which introduce numerous assumptions in order to make the problem computationally tractable. The result is that the analysis of the modified algorithm might not always correspond to its real behavior (Buczak et al., 2001).

As the aiNet is both evolutionary and self-organizing, the derivation of formal convergence and proof of stability becomes very difficult. It was therefore decided to investigate convergence empirically, so as to derive a reasonable convergence criterion for the network learning process.

4.1 THE MSE AND NETWORK SIZE

Consider the simple problem of defining an aiNet to represent a Gaussian distribution composed of 100 data points in $\Re^2$. Assume a suppression threshold $\sigma_s = 0.018$, which is not directly relevant to this analysis. Fig. 1 illustrates the relationship between the MSE, given by Eq. (1) of the aiNet cells in relation to the input data (antigens) and the number of cells in the network with respect to the iterations. Although these curves cannot be said to typically represent the network pattern of behavior (there are stochastic processes within the learning algorithm), they serve to illustrate some interesting properties of the algorithm. It can be observed that in the final iterations of the algorithm, the number of cells in the network oscillates between 13 and 16 with an approximate average of 15 units (right hand scale). It can also be noted that usually, when a unit is pruned resulting in a dimension smaller than the average dimension (15 units), the MSE tends to increase. The explanation, in this case, is quite obvious: the network is loosing its capability of representing the input data set. In contrast, when another cell is inserted into the network, it gains more potential to represent the data set and thus, it tends to start reducing the error again.

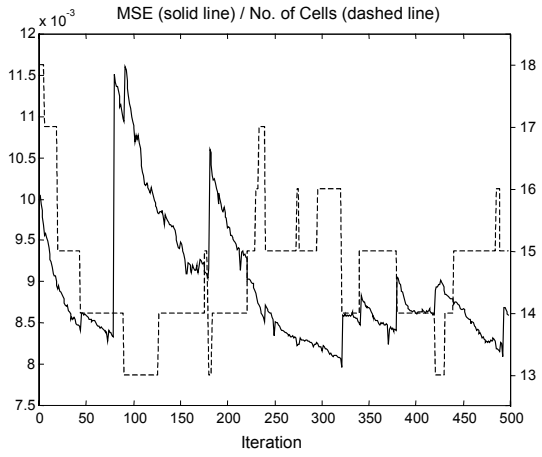


Figure 1: Relationship between the MSE given by Eq. (1) and the number of cells in the network. The left-hand axis scale is for the MSE (solid line); and the right-hand axis scale is for the

number of cells (dashed line). To facilitate the interpretation, the first two iterations of the algorithm were not depicted.

Through experimentation, it was also possible to note that the aiNet tends to stabilize around an average dimension (15 in this case).

4.2 THE ROLE OF CROSS-VALIDATION IN SUPERVISED LEARNING

Cross-validation is employed in supervised learning as an approach to reduce over fitting. Over fitting is the term used when a learning system has undergone too much training and becomes unable to generalize when a new validation set is presented (Prechelt, 1998). This behavior can be easily observed by partitioning the data set into a few subsets (e.g., learning and validation) and constantly evaluating the network performance for the validation set after a certain number of adaptation steps run with only the learning set. Fig. 2 illustrates idealized curves for the errors, where the curve represents the MSE for the learning and validation sets, with the optimum point in which to stop training being illustrated. The problem with this approach is that there is no typical behavior. Instead, real situations, even for very simple problems, are much more complex, presenting a great number of local optima, such as the MSE curve depicted in Fig. 1 for the unsupervised learning of aiNet.

In an attempt to prevent this behavior, several cross-validation stopping criteria can be proposed for supervised learning (Prechelt, 1998). For example, training can be stopped when the percentage ratio between the validation error and the smallest training error is greater than a given threshold, a procedure called 'generalization loss' (GL). Another approach is to evaluate the training error within a 'window' of k iterations. Here an evaluation between how much the average training error has increased in relation to the minimum error within this window of length k. Then the ratio between this value and GL is determined. If this ratio is greater than a given threshold, then training is stopped. In all cases, the network weight set used is the one obtained before the criterion indicated over fitting.

In the aiNet case, learning is unsupervised according to Eq. (3), but the MSE among the best matching network cells (winners) and the input data can be evaluated, as is the case with the optimization clustering algorithms. As illustrated in Fig. 1, the MSE during the learning process tends to oscillate around an average value after a number of iterations, in a way similar to the validation error for supervised learning. Still, alike supervised learning, it seems that the differences between the first and the following local optima of this function are not huge.

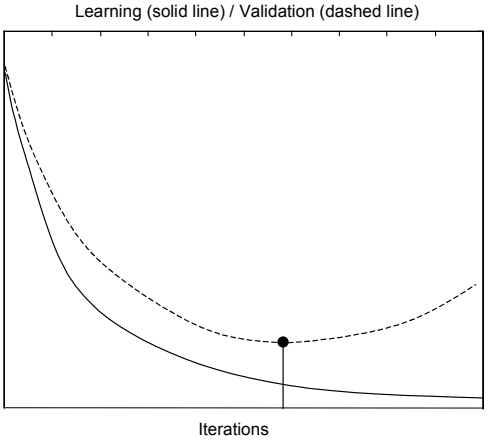


Figure 2: Idealized training and generalization error curves.

4.3 A STOPPING CRITERION FOR aiNet

Inspired by these cross-validation criteria and ideas, the following stopping criterion for the aiNet learning algorithm is proposed.

At each window of k iteration steps of aiNet, evaluate the average error (MSE_{avk}), the error at the last iteration of the window (MSE_{endk}), and the standard deviation of the error within the window (MSE_{stdk}), and store the minimal error during the whole iterative process (MSE_{opt}). Note, that as learning is unsupervised, there is a single input data set and thus, a single error is evaluated; no partition is performed in the data set. The following stopping criterion is proposed:

Stop the iterative process if
$MSE_{endk} > (MSE_{avk} + MSE_{stdk})$ after iteration t – k.

where t is the iteration (time) index. The network memory cells and their corresponding distances, $\langle \mathbf{M}_{opt}, \mathbf{D}_{opt} \rangle$, are those that lead to the smallest error (MSE_{opt}) during all the learning process up to the stopping iteration.

The use of this stopping criterion can be justified as follows. If a normal distribution for the error behavior close to its stationary state is assumed (the error can increase or decrease in short amounts), then an error exceeding one standard deviation above the average error within a window k may be regarded as a significant increase in the error. In numerical terms, a normal distribution has approximately 68% of its observations within one standard deviation of its mean. Thus, if the error at the end of the window k is larger than one standard deviation of the mean value of the window, it corresponds to an error rate of approximately 68% larger than the average error of the window k: this can be said to represent a significant increase in error.

This observation suggests that the aiNet representation capability of the input data is decreasing significantly, and usually happens when the network learning capability is close to its maximum (given its current dimension). Usually, if a novel cell is inserted into, or pruned from, the

network, then aiNet will have to reorganize the attribute values of its cells in order to account for the modified network structure. Also, the strong deterministic selective procedure adopted (k-nearest neighbors in relation to the input datum) together with the guided mutation rate, promote a greedy search around each network cell. This can be compared to a gradient ascent (hill-climbing) search (Salomon, 1998), and is a process that tends to be monotonical as far as an appropriate learning rate and network dimension are chosen. In aiNet, the learning rate and network dimensions are dynamically adjusted, and the search tends to perform steps in the opposite direction of the gradient of the MSE among the best matching cells and the given input datum.

This proposed stopping criterion has the property that it may be slow to interrupt the iterative learning: this will depend on how the learning error is oscillating. aiNet will only stop when the network is significantly loosing its capability of representing the data. However, it has the advantage that it is local to the window k, but not 'too local'; meaning that it allows the learning algorithm to go out of some local optima and to look for more 'global optima' in which to stop the iterative process. Certainly, the 'brevity' of the determined optima will depend upon the length k of the window. Another advantage of the process is that it looks for the best set of cells up to the stopping iteration.

As with the cross-validation procedures, this method also does not guarantee the convergence of the algorithm, therefore a maximum number of iteration steps must be defined. Another feature of this approach is that it assumes that the network learning, though asymptotic close to the optimal performance, suffers abrupt changes. These are consequences of the network pruning/growing of cells. If the performance of the algorithm were realistically asymptotic, then the convergence criterion would never be met, as it requires a considerable variation in the MSE evaluated. Indeed, this oscillating behavior was observed in almost all experiments performed, making the application of the proposed algorithm feasible.

5 A HIERARCHY OF aiNets

The method proposed in this section aims at enhancing the clustering capability of aiNet, such that sub-clusters can be detected within a previously defined cluster. It has also been found that this approach alleviates the problem of setting up the initial values for some user-defined learning parameters. The algorithm is presented first, and then its basic functioning is discussed.

The hierarchical algorithm for the aiNet operates as follows:

1. *Parameter definition*: define the initial values for the relevant aiNet parameters σ_s (suppression threshold) and r (inconsistency *ratio*), and set up the decaying rate $0 < \alpha < 1$ for these parameters;
2. *aiNet learning*: run the aiNet learning algorithm (Section 3) with the given parameters. The aiNet is said to have converged after it meets the stopping criterion proposed in Section 4. Note that, despite the window size $k = 20$, this criterion does not require the definition of any parameter by the user;
3. *Tree branching*: each cluster detected by the MST constructed from the resultant aiNet, generates an offspring aiNet in the next level (or depth) of networks, i.e. a new branch of the tree;
4. *Parameters updating*: reduce σ_s and r, e.g. by geometrically decreasing them by the factor α;
5. *Offspring network evaluation*: run the offspring network (new branch of the tree) for the updated (reduced) parameters; each offspring network is run with only the input data that it classifies. This means that at each level (depth) of the tree of networks, the aiNet responsible for classifying a given portion of the data set is only subjected to these data;
6. *Tree convergence*: if the offspring network does not detect a novel cluster, the process is halted, and the tree of networks is completed. Each branch of the tree represents a cluster and a sequence of branches represents the hierarchy inherent to the data mapped into these clusters. Else, while a given offspring network (branch) of the tree is still capable of idenfying more than one cluster, return to Step 4 and the process continues until no new cluster can be identified.

The behavior of the algorithm can be explained as follows. The whole input data set is presented to the first network, named *root network*. At the end of the learning process, the root network is capable of identifying the most relevant components of the data set by eliminating the redundant ones and positioning the network cells in the appropriate locations of the space. The MST method is then applied with the ratio r of that network level (root), and a certain number of clusters are identified. Each detected cluster generates a new network at the same level, branched in the root network. The input data is then mapped into each of these networks and used to evaluate their capabilities of detecting novel clusters within the previously identified clusters (sub-networks). The suppression threshold (σ_s) and inconsistency ratio (r) are reduced by the factor α. By reducing (cooling) σ_s the learning process of the offspring network becomes more accurate in relation to its input data, i.e., each input datum is represented more accuralty by a network cell or group of cells. The reduction of σ_s forces the MSE among the input data and the best matching cells to reduce. The reduction (cooling) of r allows the network to draw out less apparent clusters in the given sub-set of data. After reducing both parameters, the aiNet learning algorithm is run for each new branch (sub-network) of the tree, until no offspring networks are generated.

6 AN ILLUSTRATIVE EXAMPLE

In order to illustrate the performance of the proposed stopping criterion and hierarchical method, the algorithm was applied to a multi-structured bi-dimensional artificial data set. As originally proposed (de Castro & Von Zuben, 2000), the aiNet learning algorithm requires several parameters to be set up by the user. However, in later work, the authors demonstrated empirically that the majority of these parameters influence mainly the convergence time of the algorithm, not the final network classification. The following parameters (used in most of their simulations) were adopted for the experiment described here: $n = 4$, $d = 10$, $r = 2.0$ and $\xi = 10\%$. For a full description of the standard aiNet algorithm and its training parameters, the reader is invited to refer to (de Castro & Von Zuben, 2001).

This multi-structured data was used to evaluate a hierarchical self-organizing map (Costa & Netto, 2001). The data set illustrated in Fig. 3, is used to demonstrate that even in the presence of very different pattern structures, the proposed method is capable of detecting and separating clusters and, also, of indicating the intrinsic cluster hierarchy. The number of patterns in each class (1 to 8) is 314, 100, 100, 100, 100, 10, 53 and 57 respectively. Classes 2 to 5 were generated by multivariate Gaussian probability distributions. Class 2 is completely enclosed by a circular cluster (class 1). Class 1 is connected to class 5 by a bridge (class 6) that is a chain of intermediate objects.

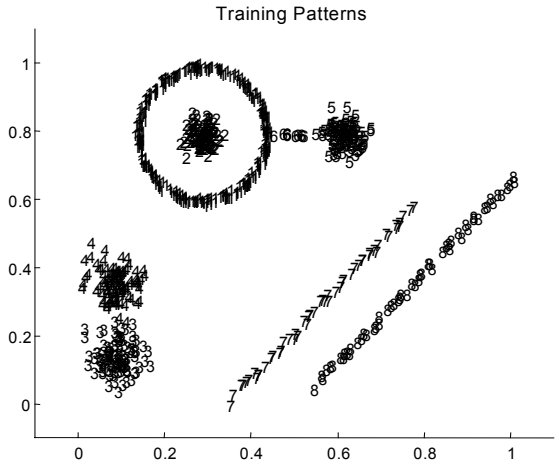


Figure 3: Multi-structured benchmark task. Discrete data set with labelled input classes. (The input data is taken to be unlabelled.)

Although the data set is labelled in the picture, the aiNet learning is unsupervised, thus the networks are presented with the unlabeled data. To run the networks, the following initial value for the suppression threshold was choosen: $\sigma_s = 0.02$. The parameters σ_s and r are decreased geometrically by the factor $\alpha = 0.9$ at each time a new level of offspring networks (tree depth) is generated. Table 1 summarizes the results obtained for this problem and Fig. 5 in the Appendix depicts the tree of networks generated, indicating the aiNet hierarchy and thus the hierarchy contained in the data set.

Table 1 presents the cooling schedule of the suppression threshold (σ_s) and inconsistency ratio (r) along with the percentage relative compression rate (CR) of each network – CR = (n. of input data)/(n. of aiNet cells), the final number of memory cells (m), the number of iterations for convergence (N_{it}), and the total number of iterations performed (TN_{it}). It can be noted from this table, that even if all the dimensions of all networks (all branches of the tree) were added together, the whole tree of networks has a final dimension of 457 cells, corresponding to a final compression rate $CR = 45.20\%$. Note also, that the number of iterations for convergence (N_{it}) is close to the total number of iterations performed (TN_{it}). The MSE corresponds to the error described in Eq. (1). Therefore, the small values for the MSE, as presented in Table 1, suggest that the network cells are a good representation for the input data.

Table 1: Percentage compression rate (CR) for each network relative to the size of the respective input data set; number of iterations for convergence of each network (N_{it}); total number of iterations simulated to achieve convergence of the stopping criterion (TN_{it}); final number of memory cells (m) in a branch network; suppression threshold (σ_s) and inconsistency ratio (r) at each level. See labels in Fig. 5 in Appendix.

(a) Branch levels 0 and 1.

	Tree Branch Label					
	0	**1-1**	**1-2**	**1-3**	**1-4**	**1-5**
m	157	60	20	25	15	51
CR (%)	81.8	85.9	62.3	56.1	85.0	74.5
MSE$\times 10^{-3}$	9.0	8.5	8.3	6.3	7.9	8.3
N_{it}	107	83	58	260	80	222
TN_{it}	120	100	60	280	140	240
σ_s	0.02			0.018		
r	2.0			1.80		

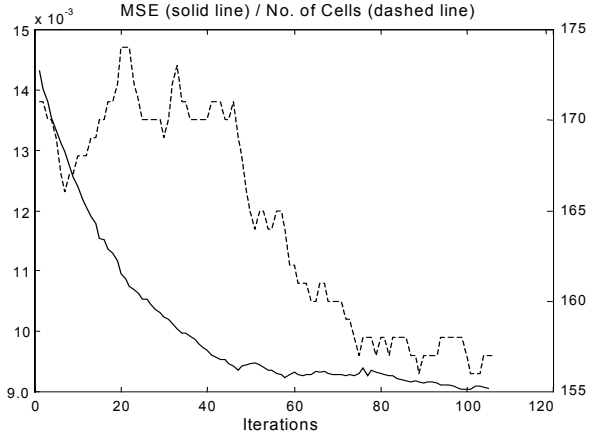
(b) Branch level 2.

	Tree Branch Label			
	2-1	**2-2**	**2-3**	**2-4**
m	28	28	23	50
CR (%)	72.55	71.13	77.00	84.57
MSE$\times 10^{-3}$	8.56	7.89	6.96	7.02
N_{it}	99	51	113	119
TN_{it}	100	60	120	160
σ_s	0.0162			
r	1.62			

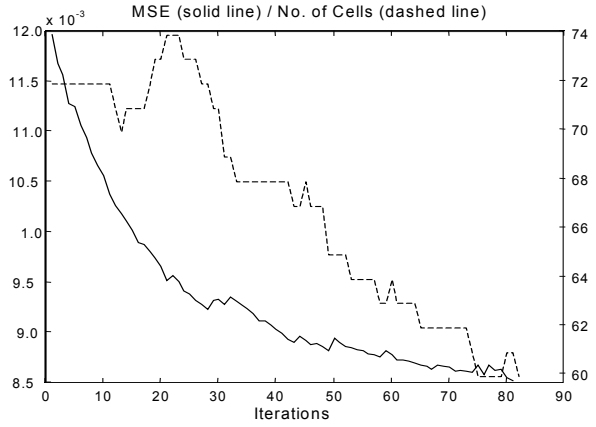
The results presented in Fig. 5 reveal interesting properties of the hierarchical tree of networks. It shows primarily that the method performs a separation of the clusters

whose inter-distances are largest, resulting in the following clusters: ([2], [1,5,6], [7], [3,4], [8]). After detecting these most distant clusters, the network learns with only the input data mapped into these clusters. The cooling of the suppression threshold and inconsistency ratio allowed the networks, in the next level of the tree, to look for less apparent dissimilarities within the previously defined clusters. It was interesting to notice that the algorithm was capable of separating cluster 6 from cluster 5, a task difficult even for a human observer. In the case of aiNet, this was only possible because the algorithm generated a reduced number of cells to represent cluster 6 (only two cells), which were significantly far apart from cluster 5. Furthermore, it can be noticed that the hierarchical patterns discovered by this algorithm, are similar to the perception that human observers might have. For example, we first identify the larger differences among clusters, and then we 'focus' our attention on minor details. Note also, that at each level of the network, an aiNet with an increased number of cells is generated to represent a cluster. Finally, it was observed that the network labelled 1-2 detected an element disconnected from cluster 5, what might be interpreted as an outlier.

Fig. 4 depicts the evolution of the MSE and the number of cells for the root and network 1-2, in the tree of Fig. 5, up to the convergence iteration. Note that the convergence criterion is capable of selecting the stopping iteration when the MSE appears to be in a stationary state; the MSE assumes a nearly asymptotical behavior. It can be inferred from the characteristics of the stopping criterion proposed, that the following iterations of the process were characterized by a dramatic increase in the MSE, thus promoting the end of the iterative process.



(a)



(b)

Figure 4: MSE (left-hand scale, solid curve) and number of cells (right-hand scale, dashed curve) for the root network (a) and network 1-2 of the tree depicted in Fig. 5(b).

7 CONCLUDING REMARKS

This paper proposed a stopping criterion for the artificial immune network (aiNet) model, previously introduced by de Castro and Von Zuben (2000). In addition, a hierarchical approach to training aiNet was proposed. This hierarchical method was empirically demonstrated to alleviate the problem of setting up some user-defined parameters and to allow the detection of inherent hierarchies within the input data set. It is important to stress that the hierarchical approach presented here is interesting not only in the artificial immune network context, but also from a graph-theoretical perspective. This is because the method describes how the aiNet and MST algorithms can be combined in order to identify hierarchies within a data set.

Through the use of local information concerning the network representation capability, it was possible to propose a formal stopping criterion for the network learning process that reduces the number of user-defined parameters. It is also interesting to note that the proposed hierarchical approach for the aiNet performs a sort of breadth-first search (Russell & Norvig, 1995). Each cluster detected in a given network generates an offspring network that is expanded (generates other offsprings) until no more offspring (cluster) can be detected.

The practical applications of the strategies proposed in this work are various. With larger and larger datasets being produced, as in the area of biology for example, it is important to devise alternative methods capable of analyzing large volumes of data in an automatic and unsupervised form. In particular, the use of hierarchical structures, when dealing with large amounts of data, becomes very important.

The result presented is a good indicator that the method has strong potentialities to find sub-clusters within previously defined clusters. As natural further extensions of this work, we can stress the application of the methods

(stopping criterion and hierarchical approach) to real-world problems and their comparison with hierarchical self-organizing feature maps, such as the ones used as inspiration for the development of this algorithm. Also, investigation into the development of a suitable visualization tool for a trained aiNet, which is independent upon the dimension of the input data set, should be pursued. Work presented in Timmis (2000) utilized graph drawing techniques, such as spring embedded layout to display the evolved immune networks. This may be an interesting place to start, but when dealing with larger size data, work in Mutton and Rodgers (2002), where graph theory techniques are used for efficient pre-processing of graphs and their subsequent visualization, may prove to be a fruitful avenue of investigation.

Acknowledgments

Leandro Nunes de Castro thanks CNPq (Profix, Proc. n. 540396/01-0) for the financial support.

References

Adams, R. G., Butchart, K. & Davey, N. (1999), "Hierarchical Classification with a Competitive Evolutionary Neural Tree", *Neural Networks*, **12**, pp. 541-551.

Buczak, A. L., Wang, H. H., Darabi, H. & Jafari, M. A. (2001), "Genetic Algorithm Convergence Study for Sensor Network Optimization", *Information Sciences*, **133**, pp. 267-282.

Burnet, F. M. (1959), *The Clonal Selection Theory of Acquired Immunity*, Cambridge University Press.

Cho S.-B. (1997), "Self-Organizing Map with Dynamical Node Splitting: Application to Handwritten Digit Recognition", *Neural Computation*, **9**, pp. 1345-1355.

Costa, J. A. F. & Netto, M. L. A. (1999), "Automatic Data Classification by a Hierarchy of Self-Organizing Maps", *Proc. of IEEE SMC*, **5**, pp. 419-424.

Costa, J. A. F. & Netto, M. L. A. (2001), "Clustering of Complex Shaped Data Sets via Kohonen Maps and Mathematical Morphology", In: *Data Mining and Knowledge Discovery: Theory, Tools, and Technology III*, B. Dasarathy (ed.). *Proc. of SPIE*, **4384**.

Dasgupta, D. (ed.) (1999), *Artificial Immune Systems and Their Applications*, Springer-Verlag.

de Castro, L. N. & Von Zuben, F. J. (1999), "An Improving Pruning Technique with Restart for the Kohonen Self-Organizing Feature Map", *Proc. of the IJCNN*, **3**, pp. 1916-1919.

de Castro, L. N. & Timmis, J. (2002), *Artificial Immune Systems: A New Computational Intelligence Approach*, Springer-Verlag, London.

de Castro, L. N. & Von Zuben, F. J. (2000), "An Evolutionary Immune Network for Data Clustering", *Proc. of the IEEE Brazilian Symposium on Neural Networks*, pp. 84-89.

de Castro, L. N. & Von Zuben, F. J. (2001), "aiNet: An Artificial Immune Network for Data Analysis", in *Data Mining: A Heuristic Approach*, Hussein A. Abbass, Ruhul A. Sarker, and Charles S. Newton (eds.), Idea Group Publishing, USA, Chapter XII, pp. 231-259.

de Castro, L. N. & Von Zuben, F. J. (2002), "Learning and Optimization Using the Clonal Selection Principle", accepted for publication at the *IEEE Transaction on Evolutionary Computation*, Special Issue on Artificial Immune Systems (in print).

Erwin, E., Obermayer, K. & Schulten, K. (1992), "Self-Organizing Maps: Ordering, Convergence Properties and Energy Functions", *Biol. Cybern.*, **67**, pp. 47-55.

Everitt, B. (1993), *Cluster Analysis*, Heinemann Educational Books.

Fahlman, S. E. & Lebiere, C. (1990), "The Cascade-Correlation Learning Architecture, In: *Advances in Neural Information Processing Systems*, **2**, D. S. Touretzky (ed.), Morgan Kaufmann, San Mateo, pp. 524-532."

Fritzke B. (1994), "Growing Cell Structures – A Self-Organizing Network for Unsupervised and Supervised Learning", *Neural Networks*, 7(9), pp. 1441-1460.

Jain, A. K., Murty, M. N. & Flynn, P. J. (1999), "Data Clustering: A Review", *Computing Surveys* **31**(3), pp. 264-323.

Jerne, N. K. (1974), "Towards a Network Theory of the Immune System", *Ann. Immunol.* (Inst. Pasteur) **125C**, pp. 373-389.

Kohonen T. (1982), "Self-Organized Formation of Topologically Correct Feature Maps", *Biol. Cybern.*, **43**, pp. 59-69.

Mutton, P & Rodgers, P (2002). "Spring Embedder for Preprocessing for WWW Visualization". In *Proceedings Information Visualization 2002*. IVS, IEEE..

Prechelt L. (1998**),** "Automatic Early Stopping Using Cross Validation: Quantifying the Criteria", *Neural Networks*, **11(**4), pp. 761-767.

Ritter, H. & Schulten, K. (1988), "Convergence Properties of Kohonen's Topology Conserving Maps: Fluctuations, Stability, and Dimension Selection", *Biol. Cybern.*, **60**, pp. 59-71.

Russell, S. & Norvig, P. (1995), *Artificial Intelligence: A Modern Approach*, Prentice Hall.

Solomon, R. (1998), "Evolutionary Algorithms and Gradient Search: Similarities and Differences", *IEEE Trans. on Evol. Computation*, **2**(2), pp. 45-55.

Timmis, J. (2000), *Artificial Immune Systems: A Novel Data Analysis Technique Inspired by the Immune Network Theory*, Ph.D. Dissertation, Department of Computer Science, University of Wales

Ultsch, A. (1995), "Self-Organizing Neural Networks Perform Different from Statistical k-means", *Gesellschaft für Klassification*.

Varela, F. J., Coutinho, A. Dupire, E. & Vaz, N. N. (1988), "Cognitive Networks: Immune, Neural and Otherwise", *Theoretical Immunology*, Part II, A. S. Perelson (ed.), pp. 359-375.

Zahn, C. T. (1971), "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters", *IEEE Trans. on Computers*, **C-20**(1), pp. 68-86.

Appendix

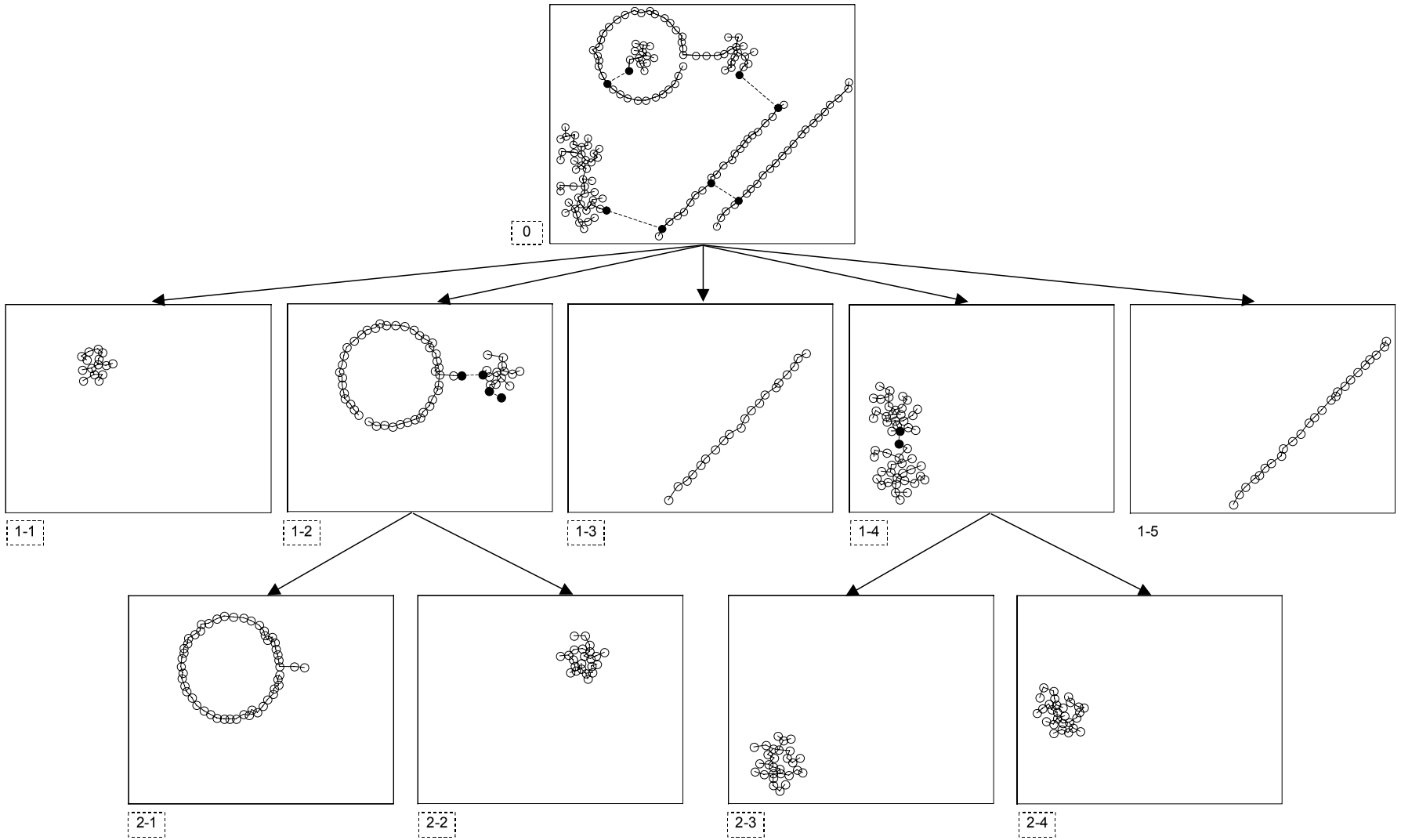


Figure 5: Hierarchy of networks. The identification of each network in the tree is represented by its level and sequence number in its bottom left part. The root network corresponds to level 0, the second layer corresponds to level 1 and so on.